



Benutzer-Leitfaden

CloudWatch Amazon-Protokolle



CloudWatch Amazon-Protokolle: Benutzer-Leitfaden

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Marken und Handelsmarken von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, die geeignet ist, Kunden irrezuführen oder Amazon in irgendeiner Weise herabzusetzen oder zu diskreditieren. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist Amazon CloudWatch Logs?	1
Features	1
Verwandte Dienste AWS	4
Preisgestaltung	5
Konzepte	5
Fakturierung und Kosten	7
Klassen protokollieren	8
Unterstützte Features	8
Erste Schritte	12
Voraussetzungen	12
Melde dich an für ein AWS-Konto	12
Einrichtung der Befehlszeilenschnittstelle	13
Verwenden Sie den vereinheitlichten CloudWatch Agenten	13
Verwenden Sie den vorherigen CloudWatch Agenten	13
CloudWatch Protokolliert die Voraussetzungen für den Agenten	14
Schnellstart: Installieren Sie den Agent auf einer ausgeführten EC2-Linux-Instance	15
Schnellstart: Installieren Sie den Agent beim Start auf einer EC2-Linux-Instance	23
Schnellstart: Verwenden Sie CloudWatch Logs mit Windows Server 2016-Instances	27
Schnellstart: Verwenden Sie CloudWatch Protokolle mit Windows Server 2012- und Windows Server 2008-Instances	39
Melden Sie den Status des CloudWatch Logs-Agenten	50
Starten Sie den CloudWatch Logs-Agent	50
Stoppen Sie den CloudWatch Logs-Agent	51
CloudWatch Referenz zum Logs-Agenten	52
Schnellstart mit CloudFormation	64
Protokollieren Sie die Aufnahme über HTTP-Endpunkte	66
Gemeinsames Verhalten	67
Bearer-Token-Authentifizierung	67
Option 1: Schnellstart mit der Konsole AWS	68
Option 2: Manuelle Einrichtung	70
Steuern Sie die Berechtigungen für die Generierung und Verwendung von CloudWatch Logs-API-Schlüsseln	73
Rotierende API-Schlüssel	75
Reaktion auf einen kompromittierten API-Schlüssel	77

Bewährte Sicherheitsmethoden für API-Schlüssel	78
Protokollierung der Verwendung von API-Schlüsseln mit CloudTrail	79
OTLP-Endpunkt	80
Anforderungsformat	80
Beispielanforderung	80
Antworten	82
OTLP-spezifisches Verhalten	82
HLC-Endpunkt	82
Eingabemodi	82
Feld „Ereignis“ (erforderlich)	83
Zeitfeld (optional)	83
Content-Type	84
Akzeptierte JSON-Werttypen	84
Endpunkt-Format	84
Anforderungsformat	85
Beispielanforderung	86
Best Practices	87
Einschränkungen	87
NDJSON-Endpunkt	88
Anforderungsformat	88
Akzeptierte JSON-Werttypen	88
Zeitstempel-Feld	90
Ungültige Zeilen	90
Beispielanforderung	91
Antworten	91
Best Practices	92
Einschränkungen	92
Strukturierter JSON-Endpunkt	93
Anforderungsformat	93
Akzeptierte JSON-Werttypen	93
Feld „Zeitstempel“	94
Beispielanforderung	95
Antworten	95
Best Practices	95
Einschränkungen	96
Vergleich der Endpunkte für die HTTP-Erfassung	97

Einen Endpunkt auswählen	98
Arbeiten mit AWS SDKs	99
Verwaltung von Protokollen	101
Erkennung und Verwaltung von Datenquellen	101
Was sind CloudWatch Logs-Datenquellen?	102
Erste Schritte	103
Systemfelder	104
Zugreifen auf Datenquellen	105
Beziehung zu Protokollgruppen	105
Funktionen, die durch Datenquellen ermöglicht werden	105
Unterstützte Datenquellen	106
Unterstützt AWS-Services für Datenquellen	106
Unterstützte Quellen von Drittanbietern für Datenquellen	111
Benutzerdefinierte Quellen	121
Logdaten mit CloudWatch Logs Insights analysieren	122
Unterstützte Abfragesprachen	125
CloudWatch Logs Insights-Abfragesprache (Logs Insights QL)	126
OpenSearch Piped Processing Language (PPL)	205
OpenSearch Strukturierte Abfragesprache (SQL)	220
Verwenden Sie natürliche Sprache, um CloudWatch Logs Insights-Abfragen zu generieren und zu aktualisieren	236
Beispielabfragen	237
Abmeldung von der Verwendung Ihrer Daten zur Serviceverbesserung	240
Unterstützte Protokolle und erkannte Felder	240
Felder in JSON-Protokollen	243
Erstellen Sie Feldindizes, um die Abfrageleistung zu verbessern und das Scanvolumen zu reduzieren	245
Syntax und Kontingente für Feldindexe	248
Erstellen Sie eine Feldindexrichtlinie auf Kontoebene	253
Erstellen Sie eine Feldindexrichtlinie auf Protokollgruppenebene	255
Protokollieren Sie die Gruppenauswahloptionen beim Erstellen einer Abfrage	256
Auswirkungen des Löschs einer Feldindexrichtlinie	257
Verwenden Sie Facetten, um Logs zu gruppieren und zu untersuchen	257
Um eine facettenbasierte Abfrage auszuführen	259
Um eine facettenbasierte Abfrage zu speichern	259
Um eine Facette auf Kontoebene zu erstellen	259

Facettenverwaltung mithilfe von APIs	260
Umliegende Logs in CloudWatch Logs Insights anzeigen	260
So funktionieren Umgebungsprotokolle	260
Umliegende Protokolle anzeigen	261
Sucht in den umliegenden Protokollen	262
Musteranalyse	263
Erste Schritte mit der Musteranalyse	264
Details zum Pattern-Befehl	267
Speichern und erneutes Ausführen von Abfragen.	267
Gespeicherte Abfragen mit Parametern verwenden	273
Abfrage zum Dashboard hinzufügen oder Abfrageergebnisse exportieren	276
Anzeigen von laufenden Abfragen oder Abfrageverlauf	277
Verschlüsseln Sie Abfrageergebnisse mit AWS Key Management Service	278
Einschränkungen	278
Schritt 1: Erstellen Sie ein AWS KMS key	279
Schritt 2: Festlegen von Berechtigungen auf dem KMS-Schlüssel	279
Schritt 3: Ordnen Sie Ihren Abfrageergebnissen einen KMS-Schlüssel zu	281
Schritt 4: Trennen Sie die Zuordnung eines Schlüssels zu den Abfrageergebnissen im Konto	281
Generieren Sie eine Zusammenfassung in natürlicher Sprache aus den CloudWatch Logs Insights-Abfrageergebnissen	282
Funktionsweise	282
Regionale Verfügbarkeit und Datenverarbeitung	282
Erste Schritte	283
Berechtigungen	283
Datenschutz	284
Automatisieren der Protokollanalyse mit geplanten Abfragen	285
Grundlegendes zu Konzepten für geplante Abfragen	285
IAM-Rollentrennung	285
Regions- und kontoübergreifende Nutzung	288
Zeitplanausdrücke und Zeitzonebehandlung	289
Wählen Sie eine Abfragesprache	290
Zielauswahl und Anwendungsfälle	291
Format und Struktur der Abfrageergebnisse	292
Referenz zum Ausdruck „Zeitplan“	293
Best Practices	298

Erste Schritte mit geplanten Abfragen	299
Eine geplante Abfrage erstellen	300
Geplante Abfragen anzeigen und verwalten	304
Den Verlauf der Ausführung geplanter Abfragen anzeigen	306
Aktualisierung einer geplanten Abfrage	307
Konfiguration von S3-Zielen für geplante Abfragen	309
Ergebnisse an einen Amazon S3 S3-Bucket im selben Konto senden	310
Ergebnisse an einen Amazon S3 S3-Bucket in einem anderen Konto senden	311
Verschlüsseln von Ergebnissen mit einem vom Kunden verwalteten Schlüssel AWS KMS ..	312
Problembehandlung bei geplanten Abfragen	314
Die Ausführung der Abfrage schlägt aufgrund von Berechtigungsfehlern fehl	315
Zeitlimit für Abfragen überschritten	315
Die Zielverarbeitung schlägt fehl	316
Ungültige Abfragefehler	316
Fehler bei der Parallelität von Abfragen	317
Erkennung von Protokollanomalien	319
Schweregrad und Priorität von Anomalien und Mustern	320
Sichtbarkeit und Zeit der Anomalie	321
Unterdrückung einer Anomalie	321
Häufig gestellte Fragen	321
Verwendung der Anomalieerkennung in Logs Insights CloudWatch	323
Aktivieren Sie die Anomalieerkennung für eine Protokollgruppe	323
Zeigt gefundene Anomalien an	325
Erstellen Sie Alarme für Protokollanomaliedetektoren	328
Von Protokollanomaliedetektoren veröffentlichte Metriken	330
Verschlüsseln Sie einen Anomaliedetektor und seine Ergebnisse mit AWS KMS	331
Einschränkungen	331
Problembehandlung mit CloudWatch Logs Live Tail	336
Starten Sie eine Live-Tail-Sitzung mit dem AWS CLI	337
nur drucken	337
interaktiv	337
Starten Sie eine Live-Tail-Sitzung in der Konsole	339
Cross-account Regionsübergreifende Protokollzentralisierung	343
Konzepte der Datenzentralisierung	343
Einrichtung der Protokollzentralisierung	345
Voraussetzungen	345

Anpassen der Namen der Zielprotokollgruppen	346
Eine Zentralisierungsregel erstellen	347
Eine Zentralisierungsregel ändern	351
Eine Zentralisierungsregel anzeigen	351
Löschen einer Zentralisierungsregel	352
Zentralisierungsregeln für Überwachung und Fehlerbehebung	352
Integritätsstatus der Regel Zentralisierung	353
Überwachung von API-Aufrufen für die Zentralisierung mit AWS CloudTrail	353
Empfehlungen zur Überwachung	354
Arbeiten mit Protokollgruppen und Protokollstreams	357
Eine Protokollgruppe erstellen	357
Protokolle an eine Protokollgruppe senden	358
Protokolldaten anzeigen	358
Suchen von Protokolldaten mithilfe von Filtermustern	359
Suchen von Protokolleinträge mithilfe der Konsole	360
Suchen Sie nach Protokolleinträgen mithilfe der AWS CLI	360
Wechseln von Metriken zu Protokollen	361
Fehlerbehebung	362
Ändern der Protokolldaten-Aufbewahrung	362
Protokollgruppen vor dem Löschen schützen	363
Protokollgruppen vor dem Löschen schützen	363
Löschschutz aktivieren	363
Markieren von Protokollgruppen	364
Grundlagen zu Tags (Markierungen)	365
Verfolgen der Kosten mithilfe von Markierungen	366
Tag-Einschränkungen	366
Taggen von Protokollgruppen mit dem AWS CLI	367
Taggen von Protokollgruppen mithilfe der CloudWatch Logs-API	367
Verschlüsseln Sie Protokolldaten mit AWS KMS	368
Einschränkungen	369
Schritt 1: Erstellen Sie einen AWS KMS Schlüssel	279
Schritt 2: Festlegen von Berechtigungen auf dem KMS-Schlüssel	370
Schritt 3: Legen Sie die Berechtigungen für den aufrufenden IAM-Prinzipal fest	373
Schritt 4: Ordnen Sie einen KMS-Schlüssel einer Protokollgruppe zu	335
Schritt 5: Trennen Sie die Zuordnung des Schlüssels zu einer Protokollgruppe	335
KMS-Schlüssel und Verschlüsselungskontext	377

Den Schutz vertraulicher Protokolldaten mit Maskierung unterstützen	380
Informationen über Datenschutzrichtlinien	384
Erforderliche IAM-Berechtigungen zum Erstellen oder Arbeiten mit einer Datenschutzrichtlinie	386
Erstellen einer kontoweiten Datenschutzrichtlinie	391
Erstellen einer Datenschutzrichtlinie für eine einzelne Protokollgruppe	395
Unmaskierte Daten anzeigen	398
Prüfergebnisberichte	399
Arten von Daten, die Sie schützen können	400
Logs während der Aufnahme transformieren	446
Log-Transformatoren erstellen und verwalten	448
Erstellen Sie eine Transformer-Richtlinie auf Kontoebene	449
Bearbeiten oder löschen Sie eine Transformer-Richtlinie auf Kontoebene	451
Erstellen Sie einen log-group-level Log-Transformer von Grund auf	451
Erstellen Sie einen log-group-level Transformator, indem Sie einen vorhandenen kopieren .	453
Bearbeiten Sie einen log-group-level Transformator	453
Löschen Sie einen log-group-level Transformator	454
Konfigurierbare Prozessoren vom Typ Parser	455
parseJSON	455
Grok	457
csv	490
parseKeyValue	492
Integrierte Prozessoren für AWS verkaufte Logs	494
Analysieren Sie WAF	495
Analysieren Sie PostGres	497
Analysieren Sie CloudFront	498
ParseRoute53	500
ParseVPC	501
Parse nach OCSF	502
Prozessoren mutieren durch Zeichenketten	504
lowerCaseString	504
upperCaseString	505
SplitString	506
SubstituteString	509
TrimString	513
JSON-Prozessoren mutieren	514

Schlüssel hinzufügen	514
Schlüssel löschen	516
MoveKeys	517
Schlüssel umbenennen	519
Wert kopieren	520
listToMap	522
Prozessoren für Datentypkonverter	528
TypeConverter	528
DateTimeConverter	530
Metriken und Fehler bei der Transformation	532
Analysieren Sie mit Amazon OpenSearch Service	533
Schritt 1: Erstellen Sie die Integration mit Service OpenSearch	535
Erforderliche Berechtigungen	536
Erstellen Sie die Integration	543
Schritt 2: Erstellen Sie Dashboards für verkaufte Logs	545
Dashboards für verkaufte Logs anzeigen, bearbeiten oder löschen	546
Dashboards für verkaufte Logs in Logs or Service anzeigen CloudWatch OpenSearch	546
Gewähren Sie zusätzlichen IAM-Rollen oder IAM-Benutzern Zugriff auf das Dashboard	546
Dashboard-Konfiguration bearbeiten	547
Lösche ein ausgeliefertes Log-Dashboard	547
Löschen Sie alle verkauften Log-Dashboard-Integrationen mit Service OpenSearch	548
IAM-Richtlinien für Benutzer	549
Berechtigungen, die für die Integration erforderlich sind	551
Greifen Sie mit der Integration von S3 Tables auf Protokolle zu	553
Die Integration von S3-Tabellen verstehen	553
Kernkomponenten	553
Datenfluss zu S3-Tabellen	553
Wann sollte die Integration von S3-Tabellen verwendet werden	554
Voraussetzungen	555
IAM-Berechtigungen	555
KMS-Schlüsselrichtlinie (für verschlüsselte Daten)	557
Erste Schritte	558
Integration von Amazon Amazon S3 S3-Tabellen mit AWS Analysedienste — Verwenden der Amazon S3 S3-Konsole (Link)	559
Lake Formation Formation-Berechtigungen konfigurieren	560
Mit Analysetools Connect	561

Metrikfilter	562
Konzepte	563
Filtermustersyntax für metrische Filter	565
Konfigurieren von Metrikwerten für einen Metrikfilter	566
Veröffentlichen von Dimensionen mit Metriken aus Protokollereignissen	567
Verwendung von Werten in Protokollereignissen zum Erhöhen des Metrikwerts	570
Erstellen von Metrikfiltern	571
Erstellen eines Metrikfilters für eine Protokollgruppe	572
Beispiel: Zählen von Protokollereignissen	573
Beispiel: Zählen des Vorkommens eines Begriffs	575
Beispiel: Zählen von HTTP-404-Codes	576
Beispiel: Zählen von HTTP-4xx-Codes	579
Beispiel: Extrahieren von Feldern aus einem Apache-Protokoll und Zuweisen von Dimensionen	581
Auflisten von Metrikfiltern	583
Löschen eines Metrikfilters	584
Abonnement-Filter	585
Konzepte	587
Abonnementfilter auf Gruppenebene protokollieren	588
Beispiel 1: Abonnementfilter mit Amazon Kinesis Data Streams	589
Beispiel 2: Abonnementfilter mit AWS Lambda	595
Beispiel 3: Abonnementfilter mit Amazon Data Firehose	599
Beispiel 4: Abonnementfilter mit Amazon OpenSearch Service	607
Abonnementfilter auf Kontoebene	608
Beispiel 1: Abonnementfilter mit Amazon Kinesis Data Streams	609
Beispiel 2: Abonnementfilter mit AWS Lambda	615
Beispiel 3: Abonnementfilter mit Amazon Data Firehose	620
Kontoübergreifende, regionsübergreifende Abonnements	628
Kontoübergreifender regionsübergreifender Austausch von Protokolldaten mithilfe von Amazon Kinesis Data Streams	629
Kontoübergreifender regionsübergreifender Austausch von Protokolldaten mit Firehose	650
Kontoübergreifende, regionsübergreifende Abonnements auf Kontoebene mit Amazon Kinesis Data Streams	665
Kontoübergreifende, regionsübergreifende Abonnements auf Kontoebene mit Firehose	684
Confused-Deputy-Prävention	697
Verhinderung der Rekursion von Protokollen	698

Filtermustersyntax	700
Unterstützte reguläre Ausdrücke	701
Begriffe mit regulären Ausdrücken abgleichen	704
Begriffe in unstrukturierten Protokollereignissen abgleichen	704
Abgleichen von Begriffen in JSON-Protokollereignissen	708
Begriffe in Leerzeichen-getrennten Protokollereignissen abgleichen	717
Protokollierung aktivieren von AWS service	722
Unterstützte Protokollziele	725
Protokollierung, für die zusätzliche Berechtigungen [V1] erforderlich sind	742
An Logs gesendete CloudWatch Protokolle	742
An Amazon S3 gesendete Protokolle	746
An Firehose gesendete Logs	751
Protokollierung, für die zusätzliche Berechtigungen [V2] erforderlich sind	753
Protokolle, die an CloudWatch Logs gesendet wurden	754
An Amazon S3 gesendete Protokolle	758
An Firehose gesendete Logs	763
Spuren wurden gesendet an X-Ray	766
Service-specific Berechtigungen	769
Console-specific Berechtigungen	770
Cross-account Beispiel für eine Zustellung	771
Lieferquelle erstellen	772
Versand an einen Amazon S3 S3-Bucket konfigurieren	772
Konfigurieren Sie die Lieferung an einen Firehose-Stream	776
Cross-service verwirrter Stellvertreter, Prävention	778
Automatisieren Sie die Aktivierung von Protokollen mit Regeln zur Telemetrieaktivierung	779
Hierarchie der Regel-Evaluierung	780
Eine Aktivierungsregel erstellen	780
Aktivieren Sie die Protokollierung aus Quellen von Drittanbietern	781
Direkte Integrationen von Drittanbietern	781
Zusätzliche Quellen über AWS Security Hub CSPM	782
Exportieren von Protokolldaten nach Amazon S3	787
Konzepte	788
Exportieren von Protokolldaten in Amazon S3 mit der Konsole	789
Export für dasselbe Konto (Konsole)	790
Kontoübergreifender Export (Konsole)	797
Exportieren Sie Protokolldaten nach Amazon S3 mit dem AWS CLI	807

Export für dasselbe Konto (CLI)	808
Kontoübergreifender Export (CLI)	815
Exportaufgaben beschreiben (CLI)	825
Eine Exportaufgabe abbrechen (CLI)	827
Daten zum OpenSearch Service streamen	828
Voraussetzungen	828
Abonnieren Sie OpenSearch Service für eine Protokollgruppe	829
Codebeispiele	831
Grundlagen	832
Aktionen	833
Szenarien	899
Amazon ECS Service Connect konfigurieren	900
Erstellen Sie Ihre erste Lambda-Funktion	916
Ausführen einer umfangreichen Abfrage	928
Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion	966
Verschlüsseln Sie Nachschlagetabellen mit AWS KMS	969
Wie CloudWatch Logs AWS KMS für Lookup-Tabellen verwendet	970
Erforderliche -Berechtigungen	970
Schritt 1: Erstellen Sie einen AWS KMS Schlüssel	971
Schritt 2: Festlegen von Berechtigungen auf dem KMS-Schlüssel	972
Schritt 3: Verknüpfen Sie einen KMS-Schlüssel mit einer Nachschlagetabelle	973
Überlegungen	974
Sicherheit	975
Datenschutz	976
Verschlüsselung im Ruhezustand	977
Verschlüsselung während der Übertragung	977
Identity and Access Management	977
Authentifizierung	978
Zugriffskontrolle	978
Übersicht über die Verwaltung von Zugriffsberechtigungen	979
Verwenden von identitätsbasierten Richtlinien (IAM-Richtlinien)	984
Compliance-Validierung	1015
Ausfallsicherheit	1016
Sicherheit der Infrastruktur	1016
Schnittstellen-VPC-Endpunkte	1017
Verfügbarkeit	1017

Einen VPC-Endpunkt für CloudWatch Logs erstellen	1017
Testen der Verbindung zwischen Ihrer VPC und Logs CloudWatch	1018
Steuern des Zugriffs auf Ihren CloudWatch Logs-VPC-Endpunkt	1019
Support für VPC-Kontextschlüssel	1020
Protokollierung von API- und Konsolenoperationen mit AWS CloudTrail	1021
Informationen zur Generierung von Abfragen in CloudTrail	1026
Grundlagen zu -Protokolldateieinträgen	1027
Überwachung der Nutzung mit CloudWatch Metriken	1029
CloudWatch Protokolliert Metriken	1029
Dimensionen für CloudWatch Logs-Metriken	1034
Metriken und Dimensionen von Log Transformer	1035
Metriken und Dimensionen der Zentralisierung	1036
CloudWatch Protokolliert Metriken zur Dienstnutzung	1040
Servicekontingente	1043
Verwaltung Ihrer CloudWatch Logs-Dienstkontingente	1055
Dokumentverlauf	1057
AWS Glossar	1076
.....	mlxxvii

Was ist Amazon CloudWatch Logs?

Sie können Amazon CloudWatch Logs verwenden, um Ihre Protokolldateien von Amazon Elastic Compute Cloud (Amazon EC2) -Instances, AWS CloudTrail Route 53 und anderen Quellen zu überwachen, zu speichern und darauf zuzugreifen.

CloudWatch Logs ermöglicht es Ihnen, die Protokolle all Ihrer Systeme, Anwendungen und AWS Dienste, die Sie verwenden, in einem einzigen, hoch skalierbaren Service zu zentralisieren. Sie können sie dann einfach anzeigen, nach bestimmten Fehlercodes oder Mustern durchsuchen, sie nach bestimmten Feldern filtern oder sie für future Analysen sicher archivieren. CloudWatch Mithilfe von Logs können Sie all Ihre Logs, unabhängig von ihrer Quelle, als einen einzigen und konsistenten Ablauf von Ereignissen betrachten, der nach Zeit geordnet ist.

CloudWatch Logs unterstützt auch das Abfragen Ihrer Protokolle mit einer leistungsstarken Abfragesprache, das Prüfen und Maskieren sensibler Daten in Protokollen und das Generieren von Metriken aus Protokollen mithilfe von Filtern oder einem eingebetteten Protokollformat.

CloudWatch Logs unterstützt zwei Protokollklassen. Protokollgruppen der Protokollklasse CloudWatch Logs Standard unterstützen alle CloudWatch Logs-Funktionen. Für Protokollgruppen der Protokollklasse CloudWatch Logs Infrequent Access fallen geringere Aufnahmegebühren an und sie unterstützen einen Teil der Funktionen der Standardklasse. Weitere Informationen finden Sie unter [Klassen protokollieren](#).

Features

- Zwei Protokollklassen aus Gründen der Flexibilität — CloudWatch Logs bietet zwei Protokollklassen, sodass Sie eine kostengünstige Option für Protokolle haben, auf die Sie selten zugreifen. Sie haben auch eine Option mit vollem Funktionsumfang für Protokolle, die eine Echtzeitüberwachung oder andere Funktionen erfordern. Weitere Informationen finden Sie unter [Klassen protokollieren](#).
- Logdaten abfragen — Mit CloudWatch Logs Insights können Sie Ihre Logdaten interaktiv suchen und analysieren. Sie können Abfragen durchführen, um effizienter und effektiver auf betriebliche Probleme zu reagieren. CloudWatch Logs Insights enthält eine speziell entwickelte Abfragesprache mit einigen einfachen, aber leistungsstarken Befehlen. Wir stellen Beispielabfragen, Befehlsbeschreibungen, automatische Abfragevervollständigung und Protokollfeldererkennung zur Verfügung, um Ihnen den Einstieg zu erleichtern. Beispielabfragen sind

für verschiedene Arten von AWS Dienstprotokollen enthalten. Um zu beginnen, sehen Sie sich [Logdaten mit CloudWatch Logs Insights analysieren](#) an.

- Erstellen Sie Feldindizes, um Abfragen effizienter zu gestalten — Sie können Feldindizes von Feldern in Ihren Protokollereignissen erstellen. Wenn Sie dann einen Feldindex in einer CloudWatch Logs Insights-Abfrage verwenden, versucht die Abfrage, die Verarbeitung von Protokollereignissen zu überspringen, von denen bekannt ist, dass sie das indizierte Feld nicht enthalten. Diese Abfrage reduziert das Scanvolumen Ihrer Abfragen, sodass Ergebnisse schneller zurückgegeben werden können. Um zu beginnen, sehen Sie sich [Erstellen Sie Feldindizes, um die Abfrageleistung zu verbessern und das Scanvolumen zu reduzieren](#) an.
- Mit Live Tail erkennen und debuggen – Sie können Live Tail verwenden, um Vorfälle schnell zu beheben, indem Sie sich eine Streamingliste mit neuen Protokollereignissen anzeigen lassen, sobald sie erfasst werden. Sie können erfasste Protokolle nahezu in Echtzeit anzeigen, filtern und hervorheben, sodass Sie Probleme schnell erkennen und lösen können. Sie können die Protokolle nach von Ihnen angegebenen Begriffen filtern und auch Protokolle hervorheben, die bestimmte Begriffe enthalten, damit Sie schnell finden, wonach Sie suchen. Weitere Informationen finden Sie unter [Problembehandlung mit CloudWatch Logs Live Tail](#).
- Protokolle von EC2 Amazon-Instances überwachen — Sie können CloudWatch Logs verwenden, um Anwendungen und Systeme anhand von Protokolldaten zu überwachen. CloudWatch Logs kann beispielsweise die Anzahl der Fehler verfolgen, die in Ihren Anwendungsprotokollen auftreten, und Ihnen eine Benachrichtigung senden, wenn die Fehlerrate einen von Ihnen festgelegten Schwellenwert überschreitet. CloudWatch Logs verwendet Ihre Protokolldaten zur Überwachung, sodass keine Codeänderungen erforderlich sind. Sie können beispielsweise Anwendungsprotokolle auf bestimmte wörtliche Begriffe (wie "NullPointerException,") hin überwachen oder die Anzahl der Vorkommen eines wörtlichen Begriffs an einer bestimmten Position in den Protokolldaten zählen (z. B. die Statuscodes „404“ in einem Apache-Zugriffsprotokoll). Wenn der Begriff, nach dem Sie suchen, gefunden wird, meldet CloudWatch Logs die Daten anhand einer von Ihnen angegebenen CloudWatch Metrik. Die Protokolldaten werden während der Übermittlung und Speicherung verschlüsselt. Um zu beginnen, sehen Sie sich [Erste Schritte mit CloudWatch Logs](#) an.
- AWS CloudTrail Protokollierte Ereignisse überwachen — Sie können Alarme in CloudWatch bestimmten API-Aktivitäten, die von erfasst wurden, einrichten CloudTrail und Benachrichtigungen erhalten und die Benachrichtigung zur Fehlerbehebung verwenden. Erste Schritte finden Sie im AWS CloudTrail Benutzerhandbuch unter [CloudTrail Ereignisse an CloudWatch Protokolle senden](#).

- Vertrauliche Daten prüfen und maskieren – Wenn Ihre Protokolle sensible Daten enthalten, können Sie sie mithilfe von Datenschutzrichtlinien schützen. Mit diesen Richtlinien können Sie die vertraulichen Daten überprüfen und maskieren. Bei aktiviertem Datenschutz werden vertrauliche Daten, die den von Ihnen ausgewählten Datenkennungen entsprechen, standardmäßig maskiert. Weitere Informationen finden Sie unter [Den Schutz vertraulicher Protokolldaten mit Maskierung unterstützen](#).
- Aufbewahrung von Protokollen – Standardmäßig werden Protokolle unbegrenzt aufbewahrt und laufen nicht ab. Sie können die Aufbewahrungsrichtlinie für jede Protokollgruppe anpassen und Protokolle entweder unbegrenzt speichern oder einen Aufbewahrungszeitraum zwischen 10 Jahren und einem Tag auswählen.
- Löschschutz — Eine Schutzmaßnahme, die das versehentliche Löschen von Protokollgruppen und ihren Protokollströmen verhindert. Wenn der Löschschutz für eine Protokollgruppe aktiviert ist, blockiert er alle Löschvorgänge, bis er ausdrücklich deaktiviert wird. Standardmäßig ist der Löschschutz nicht aktiviert. Diese optionale Funktion schützt wichtige Betriebs- und Compliance-Daten vor unbeabsichtigtem Löschen, z. B. Protokollgruppen, die Auditdaten enthalten, und Protokolle von Produktionsanwendungen zur Problembehandlung und Analyse.
- Protokolldaten archivieren — Sie können CloudWatch Logs verwenden, um Ihre Protokolldaten in einem äußerst stabilen Speicher zu speichern. Der CloudWatch Logs-Agent macht es einfach, sowohl rotierte als auch nicht rotierte Protokolldaten schnell von einem Host in den Protokolldienst zu senden. Sie können dann bei Bedarf auf die unformatierten Protokolldaten zugreifen.
- DNS-Abfragen von Route 53 protokollieren — Mithilfe von CloudWatch Logs können Sie Informationen über die DNS-Abfragen protokollieren, die Route 53 empfängt. Weitere Informationen finden Sie unter [Protokollieren von DNS-Abfragen](#) im Entwicklerhandbuch für Amazon Route 53.
- Zentralisierung von Protokollen über Konten und Regionen hinweg — Mithilfe der Zentralisierung von CloudWatch Logs können Sie konto- und regionsübergreifende Zentralisierungsregeln definieren, mit denen Protokolldaten, die in Umgebungen mit mehreren Konten und Regionen aufgenommen wurden, in eine zentrale Region und ein zentrales Konto repliziert werden. Sie können eine Backup-Region innerhalb des zentralen Kontos konfigurieren, um die Ausfallsicherheit zu erhöhen, Verschlüsselungseinstellungen für neu erstellte Protokollgruppen im zentralen Konto definieren und Metrik- und Abonnementfilter für Protokollereignisse von bestimmten Quellkonten und Regionen mit erweiterten Filterfunktionen einrichten.

Verwandte Dienste AWS

Die folgenden Dienste werden in Verbindung mit CloudWatch Logs verwendet:

- AWS CloudTrail ist ein Webservice, mit dem Sie die Aufrufe der CloudWatch Logs-API für Ihr Konto überwachen können, einschließlich der Aufrufe von AWS-Managementkonsole, AWS Command Line Interface (AWS CLI) und anderen Diensten. Wenn die CloudTrail Protokollierung aktiviert ist, werden API-Aufrufe in Ihrem Konto CloudTrail erfasst und die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket gesendet. Jede Protokolldatei kann eine oder mehrere Datensätze enthalten, je nachdem, wie viele Aktionen zur Erfüllung einer Anfrage durchgeführt werden müssen. Weitere Informationen zu AWS CloudTrail finden Sie unter [Was ist AWS CloudTrail?](#) im AWS CloudTrail Benutzerhandbuch. Ein Beispiel für den Datentyp, der in CloudTrail Protokolldateien CloudWatch geschrieben wird, finden Sie unter [Logging CloudWatch protokolliert API- und Konsolenoperationen in AWS CloudTrail](#).
- AWS Identity and Access Management (IAM) ist ein Webservice, mit dem Sie den Zugriff Ihrer Benutzer auf AWS Ressourcen sicher kontrollieren können. Kontrollieren Sie mit IAM, wer Ihre AWS -Ressourcen verwenden kann (Authentifizierung) und welche Ressourcen auf welche Weise verwendet werden können (Autorisierung). Weitere Informationen finden Sie unter [Was ist IAM?](#) im IAM-Benutzerhandbuch.
- Amazon Kinesis Data Streams ist ein Webservice, den Sie für die schnelle und kontinuierliche Aufnahme und Aggregation von Daten verwenden können. Der verwendete Datentyp umfasst Protokolldaten zur IT-Infrastruktur, Anwendungsprotokolle, Data-Feeds von sozialen Medien, Marktdaten-Feeds sowie Web-Clickstream-Daten. Da die Reaktionszeit für die Aufnahme und Verarbeitung der Daten in Echtzeit erfolgt, ist die Verarbeitung in der Regel ein leichtgewichtiger Prozess. Weitere Informationen finden Sie unter [Was ist Amazon Kinesis Data Streams?](#) im Entwicklerhandbuch für Amazon Kinesis Data Streams.
- AWS Lambda ist ein Web-Service, den Sie verwenden können, um Anwendungen zu erstellen, die schnell auf neue Informationen reagieren. Laden Sie Ihren Anwendungscode als Lambda-Funktionen hoch und Lambda führt Ihren Code auf einer hochverfügbaren Datenverarbeitungsinfrastruktur aus und erledigt die gesamte Administration der Datenverarbeitungsressourcen, einschließlich der Server- und Betriebssystemwartung, Kapazitätsbereitstellung, automatischen Skalierung, Bereitstellung von Code- und Sicherheitspatches sowie der Code-Überwachung und -Protokollierung. Sie müssen lediglich Ihren Code in einer der von Lambda unterstützten Sprachen angeben. Weitere Informationen finden Sie unter [Was ist AWS Lambda?](#) im AWS Lambda Entwicklerhandbuch.

Preisgestaltung

Wenn Sie sich für Logs registrieren AWS, können Sie im Rahmen des kostenlosen [Kontingents AWS kostenlos](#) mit CloudWatch Logs beginnen.

Standardtarife gelten für Protokolle, die von anderen Services mithilfe von CloudWatch Logs gespeichert werden (z. B. Amazon VPC-Flow-Logs und Lambda-Logs).

Weitere Informationen zur Preisgestaltung finden Sie unter [CloudWatch Amazon-Preise](#).

Weitere Informationen zur Analyse Ihrer Kosten und Nutzung von CloudWatch Logs sowie CloudWatch bewährte Methoden zur Kostensenkung finden Sie unter [CloudWatch Abrechnung und Kosten](#).

Konzepte von Amazon CloudWatch Logs

Die Terminologie und Konzepte, die für Ihr Verständnis und Ihre Verwendung von CloudWatch Logs von zentraler Bedeutung sind, werden im Folgenden beschrieben.

Klasse „Protokoll“

CloudWatch Logs bietet zwei Klassen von Protokollgruppen. Die Standard-Protokollklasse ist eine Option mit vollem Funktionsumfang für Protokolle, die eine Echtzeitüberwachung erfordern, oder für Protokolle, auf die Sie häufig zugreifen. Die Protokollklasse für seltenen Zugriff ist eine kostengünstigere Option für Protokolle, auf die Sie seltener zugreifen. Sie unterstützt einen Teil der Funktionen der Standard-Protokollklasse.

Protokollereignisse

Ein Protokollereignis ist ein Datensatz von einigen Aktivitäten, der von der überwachten Anwendung oder Ressource aufgezeichnet wird. Der Protokollereignisdatsatz, den CloudWatch Logs versteht, enthält zwei Eigenschaften: den Zeitstempel, zu dem das Ereignis eingetreten ist, und die unformatierte Ereignismeldung. Ereignismeldungen müssen UTF-8-kodiert sein.

Protokollstreams

Ein Protokollstream ist eine Abfolge von Protokollereignissen, die dieselbe Quelle nutzen. Genauer gesagt ist ein Protokollstream allgemein dafür gedacht, die Abfolge der aus der überwachten Anwendungs-Instance oder Ressource stammenden Ereignisse darzustellen. Ein Protokollstream kann beispielsweise mit einem Apache-Zugriffsprotokoll auf einem bestimmten

Host verknüpft sein. Wenn Sie einen Protokollstream nicht mehr benötigen, können Sie ihn mit dem `delete-log-stream` Befehl [aws logs](#) löschen.

Protokollgruppen

Protokollgruppen definieren Gruppen von Protokollstreams, die dieselben Einstellungen für die Aufbewahrung, Überwachung und Zugriffskontrolle besitzen. Jeder Protokollstream muss zu einer Protokollgruppe gehören. Wenn Sie beispielsweise über einen separaten Protokoll-Stream für die Apache-Zugriffsprotokolle von jedem Host verfügen, können Sie diese in einer einzelnen Protokollgruppe mit dem Namen `MyWebsite.com/Apache/access_log` gruppieren.

Es gibt keine Begrenzung dazu, wie viele Protokollstreams zu einer Protokollgruppe gehören können.

Metrikfilter

Sie können Metrikfilter verwenden, um Metrik-Beobachtungen von übernommenen Ereignissen zu extrahieren und in Datenpunkte in einer CloudWatch -Metrik umzuwandeln. Metrikfilter sind Protokollgruppen zugewiesen, und alle einer Protokollgruppe zugewiesenen Filter werden auf ihre Protokollstreams angewendet.

Einstellungen für die Aufbewahrung

Mithilfe der Aufbewahrungseinstellungen kann festgelegt werden, wie lange Protokollereignisse in CloudWatch Logs aufbewahrt werden. Abgelaufene Protokollereignisse werden automatisch gelöscht. Wie Metrikfilter werden auch die Einstellungen für die Aufbewahrung den Protokollgruppen zugewiesen, und die einer Protokollgruppe zugewiesene Aufbewahrung wird auf ihre Protokollstreams angewendet.

Löschschutz

Der Löschschutz ist eine Schutzmaßnahme, die das versehentliche Löschen von Protokollgruppen und ihren Protokollströmen verhindert. Wenn der Löschschutz für eine Protokollgruppe aktiviert ist, blockiert er alle Löschvorgänge, bis er ausdrücklich deaktiviert wird. Standardmäßig ist der Löschschutz nicht aktiviert. Diese optionale Funktion schützt wichtige Betriebs- und Compliance-Daten vor unbeabsichtigtem Löschen, z. B. Protokollgruppen, die Auditdaten enthalten, und Protokolle von Produktionsanwendungen zur Problembehandlung und Analyse.

Amazon CloudWatch Logs Abrechnung und Kosten

Ausführliche Informationen zur Analyse Ihrer Kosten und Nutzung von CloudWatch Logs sowie bewährte Methoden zur Kostensenkung finden Sie unter [CloudWatch Abrechnung und Kosten](#). CloudWatch

Weitere Informationen zur Preisgestaltung finden Sie unter [CloudWatch Amazon-Preise](#).

Wenn Sie sich für Logs registrieren AWS, können Sie im Rahmen des kostenlosen [Kontingents AWS kostenlos](#) mit CloudWatch Logs beginnen.

Standardtarife gelten für Protokolle, die von anderen Services mithilfe von CloudWatch Logs gespeichert werden (z. B. Amazon VPC-Flow-Logs und Lambda-Logs).

Klassen protokollieren

CloudWatch Logs bietet zwei Klassen von Protokollgruppen:

- Die Protokollklasse CloudWatch Logs Standard ist eine Option mit vollem Funktionsumfang für Protokolle, die eine Echtzeitüberwachung erfordern, oder für Protokolle, auf die Sie häufig zugreifen.
- Die Protokollklasse CloudWatch Logs Infrequent Access ist eine neue Protokollklasse, mit der Sie Ihre Protokolle kostengünstig konsolidieren können. Diese Protokollklasse bietet eine Untergruppe von CloudWatch Protokollfunktionen, darunter verwaltete Erfassung, Speicherung, kontenübergreifende Protokollanalyse und Verschlüsselung zu einem niedrigeren Aufnahmepreis pro GB. Die Protokollklasse für seltenen Zugriff eignet sich ideal für Ad-hoc-Abfragen und forensische Analysen von Protokollen, auf die selten zugegriffen wird. after-the-fact

Note

Hinsichtlich der Gebühren unterscheiden sich die Protokollklassen „Standard“ und „Seltener Zugriff“ lediglich in Bezug auf die Aufnahmekosten. Die Gebühren für Speicher und CloudWatch Logs Insights sind in jeder Protokollklasse gleich.

Weitere Informationen zu den Preisen für CloudWatch Logs finden Sie unter [CloudWatch Amazon-Preise](#).

Important

Nachdem eine Protokollgruppe erstellt wurde, kann ihre Protokollklasse nicht mehr geändert werden.

Unterstützte Features

In der folgenden Tabelle sind die Funktionen für jede Protokollklasse aufgeführt.

Feature	Standard	Seltener Zugriff	
Vollständig verwaltete Protokollaufnahme und Speicherung	Ja ✓	Ja ✓	
Kontoübergreifende Funktionen	Ja ✓	Ja ✓	
Verschlüsselung mit AWS KMS	Ja ✓	Ja ✓	
CloudWatch Logs Insights-Abfragebefehle	Ja ✓	Ja ✓ (Die meisten Befehle — siehe Logs Insights QL-Befehle werden in Protokollklassen unterstützt.)	
CloudWatch Logs Insights entdeckte Felder	Ja ✓	Ja ✓	
Facetten	Ja ✓	Nein	
Verwenden von CloudWatch Pipeline zur Transformation von Protokollen	Ja ✓	Ja ✓	
Nach Amazon S3 exportieren	Ja ✓	Ja ✓	
Integration von S3-Tabellen	Ja ✓	Ja ✓	
Geplante Abfragen	Ja ✓	Ja ✓	
Verwenden von OpenSearch PPL oder OpenSearch SQL für Abfragen in CloudWatch Logs Insights;	Ja ✓	Ja ✓	

Feature	Standard	Seltener Zugriff	
Unterstützung bei Abfragen in natürlicher Sprache	Ja ✓	Nein	
CloudWatch Protokolliert die Erkennung von Anomalien	Ja ✓	Nein	
Live-Schwanz	Ja ✓	Nein	
Indizierung von Feldern	Ja ✓	Nein	
Mit dem vorherigen Zeitraum vergleichen	Ja ✓	Nein	
Abonnement-Filter	Ja ✓	Nein	
GetLogEvents und FilterLogEvents API-Operationen	Ja ✓	Nicht unterstützt Verwenden Sie CloudWatch Logs Insights, um Protokollereignisse anzuzeigen, die in Protokollgruppen der Protokollklasse für seltenen Zugriff gespeichert sind.	
Metrische Filter	Ja ✓	Nein	
Aufnahme von Container-Insights-Protokollen	Ja ✓	Nein	

Feature	Standard	Seltener Zugriff	
Erfassung von Lambda Insights-Protokollen	Ja ✓	Nein	
Schutz sensibler Daten mit Maskierung	Ja ✓	Ja ✓	
Format für eingebettete Metriken	Ja ✓	Nein	

Note

Zusätzlich zu diesen beiden Protokollklassen gibt es eine Delivery Protokollklasse. Verwenden Sie die Delivery Protokollklasse nur für die Bereitstellung von AWS Lambda Protokollen zum Speichern in Amazon S3 oder Amazon Data Firehose. Protokollereignisse in Protokollgruppen der Klasse Delivery werden zwei Tage lang gespeichert. CloudWatch Logs Diese Aufbewahrungsfrist ist festgelegt und kann nicht geändert werden. Diese Protokollklasse bietet keine umfangreichen CloudWatch Logs Funktionen wie CloudWatch Logs Insights-Abfragen.

Erste Schritte mit CloudWatch Logs

Verwenden Sie den Unified CloudWatch Agent, um Protokolle von Ihren Amazon EC2 EC2-Instances und lokalen Servern in CloudWatch Logs zu sammeln. Dieser Agent ermöglicht das Erfassen von Protokollen und erweiterte Metriken mit nur einem Agenten. Er unterstützt mehrere Betriebssysteme, einschließlich Servern mit Windows Server. Dieser Agenten ist außerdem schneller.

Wenn Sie den Unified CloudWatch Agent zur Erfassung von CloudWatch Metriken verwenden, ermöglicht er die Erfassung zusätzlicher Systemmetriken, sodass der Gast mehr Transparenz erhält. Er unterstützt auch das Erfassen von benutzerdefinierten Metriken mithilfe von StatsD oder collectd.

Weitere Informationen finden Sie unter [Installation des CloudWatch Agenten](#) im CloudWatch Amazon-Benutzerhandbuch.

Der ältere CloudWatch Logs-Agent, der nur die Erfassung von Protokollen von Linux-Servern unterstützt, ist veraltet und wird nicht mehr unterstützt. Informationen zur Migration vom älteren CloudWatch Logs Agent zum Unified Agent finden Sie unter [Erstellen der CloudWatch Agent-Konfigurationsdatei mit dem](#) Assistenten.

Inhalt

- [Voraussetzungen](#)
- [Verwenden Sie den Unified CloudWatch Agent, um mit CloudWatch Logs zu beginnen](#)
- [Verwenden Sie den vorherigen CloudWatch Agenten, um mit CloudWatch Logs zu beginnen](#)
- [Schnellstart: Verwenden Sie diese Option CloudFormation , um mit CloudWatch Logs zu beginnen](#)

Voraussetzungen

Um Amazon CloudWatch Logs verwenden zu können, benötigen Sie ein AWS Konto. Ihr AWS Konto ermöglicht es Ihnen, mithilfe von Diensten (z. B. Amazon EC2) Protokolle zu generieren, die Sie in der CloudWatch Konsole, einer webbasierten Oberfläche, einsehen können. Darüber hinaus können Sie das AWS Command Line Interface (AWS CLI) installieren und konfigurieren.

Melde dich an für ein AWS-Konto

Um loszulegen AWS, benötigen Sie eine AWS-Konto. Informationen zum Erstellen eines AWS-Konto finden Sie unter [Erste Schritte mit einem AWS-Konto](#) im AWS -Kontenverwaltung Referenzhandbuch.

Einrichtung der Befehlszeilenschnittstelle

Sie können den verwenden AWS CLI , um CloudWatch Log-Operationen durchzuführen.

Informationen zur Installation und Konfiguration von finden Sie unter [Getting Up with the AWS Command Line Interface](#) im AWS Command Line Interface Benutzerhandbuch. AWS CLI

Verwenden Sie den Unified CloudWatch Agent, um mit CloudWatch Logs zu beginnen

Weitere Informationen zur Verwendung des vereinheitlichten CloudWatch Agenten für die ersten Schritte mit CloudWatch Logs finden Sie unter [Erfassung von Metriken und Protokollen von Amazon EC2 EC2-Instances und lokalen Servern mit dem CloudWatch Agenten im CloudWatch Amazon-Benutzerhandbuch](#). Führen Sie die in diesem Abschnitt aufgeführten Schritte aus, um den Agenten zu installieren, zu konfigurieren und zu starten. Wenn Sie den Agenten nicht auch zum Sammeln von CloudWatch Metriken verwenden, können Sie alle Abschnitte ignorieren, die sich auf Metriken beziehen.

Wenn Sie derzeit den älteren CloudWatch Logs-Agenten verwenden und auf den neuen Unified Agent umsteigen möchten, empfehlen wir Ihnen, den Assistenten zu verwenden, der im neuen Agentenpaket enthalten ist. Dieser Assistent kann Ihre aktuelle CloudWatch Logs-Agent-Konfigurationsdatei lesen und den CloudWatch Agenten so einrichten, dass er dieselben Logs sammelt. Weitere Informationen zum Assistenten finden Sie unter [Create the CloudWatch Agent Configuration File with the Wizard](#) im CloudWatch Amazon-Benutzerhandbuch.

Verwenden Sie den vorherigen CloudWatch Agenten, um mit CloudWatch Logs zu beginnen

Important

CloudWatch enthält einen einheitlichen CloudWatch Agenten, der sowohl Protokolle als auch Metriken von EC2-Instances und lokalen Servern sammeln kann. Der ältere, reine Protokoll-Agent ist veraltet und wird nicht mehr unterstützt

Informationen zur Migration vom älteren Agenten, der nur für Logs genutzt werden kann, zum Unified Agent finden Sie unter [Erstellen der Agent-Konfigurationsdatei mit dem CloudWatch Assistenten](#).

Im Rest dieses Abschnitts wird die Verwendung des älteren CloudWatch Logs-Agenten für Kunden erklärt, die ihn immer noch verwenden.

Mit dem CloudWatch Logs-Agenten können Sie Protokolldaten von Amazon EC2 EC2-Instances, auf denen Linux oder Windows Server ausgeführt wird, und protokollierte Ereignisse von AWS CloudTrail veröffentlichen. Wir empfehlen, stattdessen den CloudWatch Unified Agent zu verwenden, um Ihre Protokolldaten zu veröffentlichen. Weitere Informationen über den neuen Agenten finden Sie unter [Erfassung von Metriken und Protokollen von Amazon EC2 EC2-Instances und lokalen Servern mit dem CloudWatch Agenten im CloudWatch Amazon-Benutzerhandbuch](#).

Inhalt

- [CloudWatch Protokolliert die Voraussetzungen für den Agenten](#)
- [Schnellstart: Installieren und konfigurieren Sie den CloudWatch Logs-Agent auf einer laufenden EC2-Linux-Instance](#)
- [Schnellstart: Installieren und konfigurieren Sie den CloudWatch Logs-Agent beim Start auf einer EC2-Linux-Instance](#)
- [Schnellstart: Ermöglichen Sie Ihren Amazon EC2 EC2-Instances, auf denen Windows Server 2016 ausgeführt wird, mithilfe des Logs-Agenten CloudWatch Protokolle an CloudWatch Logs zu senden](#)
- [Schnellstart: Aktivieren Sie Ihre Amazon EC2 EC2-Instances, auf denen Windows Server 2012 und Windows Server 2008 ausgeführt werden, um Protokolle an Logs zu CloudWatch senden](#)
- [Melden Sie den Status des CloudWatch Logs-Agenten](#)
- [Starten Sie den CloudWatch Logs-Agent](#)
- [Stoppen Sie den CloudWatch Logs-Agenten](#)
- [CloudWatch Referenz zum Logs-Agenten](#)

CloudWatch Protokolliert die Voraussetzungen für den Agenten

Der CloudWatch Logs-Agent benötigt Python-Version 2.7, 3.0 oder 3.3 und eine der folgenden Versionen von Linux:

- Amazon Linux Version 2014.03.02 oder höher. Amazon Linux 2 wird nicht unterstützt
- Ubuntu-Serverversion 12.04, 14.04 oder 16.04
- CentOS-Version 6, 6.3, 6.4, 6.5 oder 7.0

- Version Red Hat Enterprise Linux (RHEL) 6.5 oder 7.0
- Debian 8.0

Schnellstart: Installieren und konfigurieren Sie den CloudWatch Logs-Agent auf einer laufenden EC2-Linux-Instance

Important

Der ältere Logs-Agent ist veraltet. CloudWatch enthält einen einheitlichen Agenten, der sowohl Protokolle als auch Metriken von EC2-Instances und lokalen Servern sammeln kann. Weitere Informationen finden Sie unter [Erste Schritte mit CloudWatch Logs](#).

Informationen zur Migration vom älteren CloudWatch Logs-Agent zum Unified Agent finden Sie unter [Erstellen der CloudWatch Agent-Konfigurationsdatei mit dem Assistenten](#).

Der ältere Protokoll-Agent unterstützt nur die Versionen 2.6 bis 3.5 von Python. Darüber hinaus unterstützt der ältere CloudWatch Logs-Agent den Instanz-Metadatendienst Version 2 (IMDSv2) nicht. Wenn Ihr Server verwendet IMDSv2, müssen Sie den neueren Unified Agent anstelle des älteren CloudWatch Logs-Agenten verwenden.

Im Rest dieses Abschnitts wird die Verwendung des älteren CloudWatch Logs-Agenten für Kunden erklärt, die ihn immer noch verwenden.

Tip

CloudWatch enthält einen neuen vereinheitlichten Agenten, der sowohl Protokolle als auch Metriken von EC2-Instances und lokalen Servern sammeln kann. Wenn Sie den älteren CloudWatch Logs-Agenten noch nicht verwenden, empfehlen wir Ihnen, den neueren Unified CloudWatch Agent zu verwenden. Weitere Informationen finden Sie unter [Erste Schritte mit CloudWatch Logs](#).

Darüber hinaus unterstützt der ältere Agent Version 2 (IMDSv2) nicht den Instanz-Metadatendienst. Wenn Ihr Server verwendet IMDSv2, müssen Sie den neueren Unified Agent anstelle des älteren CloudWatch Logs-Agenten verwenden.

Im Rest dieses Abschnitts wird die Verwendung des älteren CloudWatch Logs-Agenten erklärt.

Konfigurieren Sie den älteren CloudWatch Logs-Agenten auf einer laufenden EC2-Linux-Instance

Sie können das CloudWatch Logs-Agent-Installationsprogramm auf einer vorhandenen EC2-Instance verwenden, um den CloudWatch Logs-Agenten zu installieren und zu konfigurieren. Nachdem die Installation abgeschlossen ist, werden die Protokolle automatisch von der Instance zum Protokollstream geleitet, den Sie erstellen, während der Agent installiert wird. Sie erhalten vom Agenten eine Bestätigung, dass er gestartet wurde, und er wird weiterhin ausgeführt, bis Sie ihn deaktivieren.

Sie können nicht nur den Agenten verwenden, sondern auch Protokolldaten mit dem AWS CLI CloudWatch Logs SDK oder der CloudWatch Logs API veröffentlichen. Das AWS CLI eignet sich am besten für die Veröffentlichung von Daten über die Befehlszeile oder über Skripts. Das CloudWatch Logs SDK eignet sich am besten zum Veröffentlichen von Protokolldaten direkt aus Anwendungen oder zum Erstellen Ihrer eigenen Anwendung zur Protokollveröffentlichung.

Schritt 1: Konfigurieren Sie Ihre IAM-Rolle oder Ihren IAM-Benutzer für Logs CloudWatch

Der CloudWatch Logs-Agent unterstützt IAM-Rollen und -Benutzer. Verfügt Ihre Instance bereits über eine verknüpfte IAM-Rolle, stellen Sie sicher, dass Sie die nachstehende IAM-Richtlinie einbinden. Wenn Ihrer Instance noch keine IAM-Rolle zugewiesen ist, können Sie Ihre IAM-Anmeldeinformationen für die nächsten Schritte verwenden oder Sie können dieser Instance eine IAM-Rolle zuweisen. Weitere Informationen finden Sie unter [Verbinden einer IAM-Rolle mit einer Instance](#).

So konfigurieren Sie Ihre IAM-Rolle oder Ihren IAM-Benutzer für Logs CloudWatch

1. Öffnen Sie unter <https://console.aws.amazon.com/iam/> die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Rollen.
3. Wählen Sie die Rolle, indem Sie den Rollennamen auswählen (aktivieren Sie nicht das Kontrollkästchen neben dem Namen).
4. Wählen Sie Attach Policies (Richtlinien anfügen), Create Policy (Richtlinie erstellen).

Es wird eine neue Registerkarte im Browser oder ein neues Browser-Fenster geöffnet.

5. Wählen Sie die Registerkarte JSON aus und geben Sie den Text aus dem folgenden JSON-Richtliniendokument ein.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogGroup",
      "logs:CreateLogStream",
      "logs:PutLogEvents",
      "logs:DescribeLogStreams"
    ],
    "Resource": [
      "*"
    ]
  }
]
```

6. Wählen Sie, wenn Sie fertig sind, Review policy (Richtlinie überprüfen). Die Richtlinienvalidierung meldet mögliche Syntaxfehler.
7. Geben Sie auf der Seite Review Policy (Richtlinie überprüfen) unter Name einen Namen und unter Description (Beschreibung) (optional) eine Beschreibung für die Richtlinie ein, die Sie erstellen. Überprüfen Sie unter Summary die Richtlinienzusammenfassung, um die Berechtigungen einzusehen, die von Ihrer Richtlinie gewährt werden. Wählen Sie dann Create policy aus, um Ihre Eingaben zu speichern.
8. Schließen Sie die Registerkarte im Browser oder das Browser-Fenster, und gehen Sie zurück auf die Seite Add permissions (Berechtigungen hinzufügen) für Ihre Rolle. Klicken Sie erst auf Refresh (Aktualisieren), und wählen Sie dann die neue Richtlinie, um sie ihrer Rolle anzufügen.
9. Wählen Sie Richtlinie anfügen aus.

Schritt 2: CloudWatch Logs auf einer vorhandenen Amazon EC2 EC2-Instance installieren und konfigurieren

Der Prozess zur Installation des CloudWatch Logs-Agenten unterscheidet sich je nachdem, ob auf Ihrer Amazon EC2 EC2-Instance Amazon Linux, Ubuntu, CentOS oder Red Hat ausgeführt wird. Führen Sie die für die Version von Linux entsprechenden Schritte auf Ihrer Instance aus.

So installieren und konfigurieren Sie CloudWatch Logs auf einer vorhandenen Amazon Linux-Instance

Ab Amazon Linux AMI 2014.09 ist der CloudWatch Logs-Agent als RPM-Installation mit dem `awslogs`-Paket verfügbar. Bei früheren Versionen von Amazon Linux kann der Zugriff auf das `awslogs`-Paket über die Aktualisierung Ihrer Instance mit dem Befehl `sudo yum update -y` erfolgen. Wenn Sie das `awslogs`-Paket als RPM installieren, anstatt das CloudWatch Logs-Installationsprogramm zu verwenden, erhält Ihre Instance regelmäßige Paket-Updates und Patches, AWS ohne dass der Logs-Agent manuell neu installiert werden muss. CloudWatch

Warning

Aktualisieren Sie den CloudWatch Logs-Agent nicht mit der RPM-Installationsmethode, wenn Sie zuvor das Python-Skript zur Installation des Agenten verwendet haben. Dies kann zu Konfigurationsproblemen führen, die den CloudWatch Logs-Agent daran hindern, Ihre Logs an zu senden CloudWatch.

1. Herstellen einer Verbindung mit Ihrer Amazon-Linux-Instance. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

Weitere Informationen zu Verbindungsproblemen finden Sie unter [Troubleshooting Connecting to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

2. Aktualisieren Sie Ihre Amazon-Linux-Instance, um über die neuesten Änderungen in den Paket-Repositorys zu verfügen.

```
sudo yum update -y
```

3. Installieren Sie das Paket `awslogs`. Dies ist die empfohlene Methode zum Installieren von `awslogs` auf Amazon-Linux-Instances.

```
sudo yum install -y awslogs
```

4. Bearbeiten Sie die Datei `/etc/awslogs/awslogs.conf`, um die nachzuverfolgenden Protokolle zu konfigurieren. Weitere Informationen zum Bearbeiten dieser Datei finden Sie unter [CloudWatch Referenz zum Logs-Agenten](#).

5. Standardmäßig verweist `/etc/awslogs/awsccli.conf` auf die Region `us-east-1`. Um Ihre Protokolle in eine andere Region zu verschieben, bearbeiten Sie die Datei `awsccli.conf` und geben diese Region an.
6. Starten Sie den Service `awslogs`.

```
sudo service awslogs start
```

Wenn Sie Amazon Linux 2 verwenden, starten Sie den `awslogs`-Service mit dem folgenden Befehl.

```
sudo systemctl start awslogsd
```

7. (Optional) Überprüfen Sie die Datei `/var/log/awslogs.log` auf Fehler, die beim Start des Services protokolliert wurden.
8. (Optional) Führen Sie den folgenden Befehl aus, um den Service `awslogs` bei jedem Systemstart zu starten.

```
sudo chkconfig awslogs on
```

Wenn Sie Amazon Linux 2 verwenden, starten Sie den Service bei jedem Systemstart mit dem folgenden Befehl.

```
sudo systemctl enable awslogsd.service
```

9. Ihnen sollten in der CloudWatch-Konsole einen kurzen Moment nach dem Ausführen des Agenten die neu erstellte Protokollgruppe und der Protokoll-Stream angezeigt werden.

Weitere Informationen finden Sie unter [An Logs gesendete Protokolldaten anzeigen CloudWatch](#).

Um CloudWatch Logs auf einer vorhandenen Ubuntu Server-, CentOS- oder Red Hat-Instanz zu installieren und zu konfigurieren

Wenn Sie ein AMI verwenden, auf dem Ubuntu Server, CentOS oder Red Hat ausgeführt wird, verwenden Sie das folgende Verfahren, um den CloudWatch Logs-Agenten manuell auf Ihrer Instance zu installieren.

1. Stellen Sie eine Verbindung zu Ihrer EC2- Instance her. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

Weitere Informationen zu Verbindungsproblemen finden Sie unter [Troubleshooting Connecting to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

2. Führen Sie das CloudWatch Logs-Agent-Installationsprogramm mit einer von zwei Optionen aus. Sie können es direkt aus dem Internet ausführen oder die Dateien herunterladen und es eigenständig ausführen.

 Note

Wenn Sie CentOS 6.x, Red Hat 6.x oder Ubuntu 12.04 verwenden, verwenden Sie die Schritte zum Herunterladen und Ausführen des Installationsprogramms. Die direkte Installation des CloudWatch Logs-Agenten über das Internet wird auf diesen Systemen nicht unterstützt.

 Note

Führen Sie auf Ubuntu `apt-get update` aus, bevor Sie die nachstehenden Befehle ausführen.

Um es direkt aus dem Internet auszuführen, verwenden Sie die folgenden Befehle und befolgen die Anweisungen:

```
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py -O
```

```
sudo python ./awslogs-agent-setup.py --region us-east-1
```

Wenn der vorherige Befehl nicht funktioniert, versuchen Sie Folgendes:

```
sudo python3 ./awslogs-agent-setup.py --region us-east-1
```

Um es herunterzuladen und eigenständig auszuführen, verwenden Sie die folgenden Befehle und die Anweisungen:

```
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-  
setup.py -O
```

```
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/  
AgentDependencies.tar.gz -O
```

```
tar xvf AgentDependencies.tar.gz -C /tmp/
```

```
sudo python ./awslogs-agent-setup.py --region us-east-1 --dependency-path /tmp/  
AgentDependencies
```

Sie können den CloudWatch Logs-Agenten installieren, indem Sie die Regionen us-east-1, us-west-1, us-west-2, ap-south-1, ap-northeast-2, ap-southeast-1, ap-southeast-2, ap-northeast-1, eu-central-1, eu-west-1 oder sa-east-1 angeben.

 Note

Weitere Informationen zur aktuellen Version und zum Versionsverlauf von `awslogs-agent-setup` finden Sie in der Datei [CHANGELOG.txt](#).

Das Installationsprogramm für den Logs Agent benötigt während der Installation bestimmte Informationen. CloudWatch Bevor Sie beginnen, müssen Sie wissen, welche Protokolldatei zu überwachen ist und müssen dessen Zeitstempelformat kennen. Darüber hinaus sollten Sie auch die folgenden Informationen bereit halten.

Item	Description
AWS Zugriffsschlüssel-ID	Drücken Sie die Eingabetaste, wenn Sie eine IAM-Rolle verwenden . Andernfalls geben Sie Ihre AWS Zugangsschlüssel-ID ein.
AWS geheimer Zugangsschlüssel	Drücken Sie die Eingabetaste, wenn Sie eine IAM-Rolle verwenden . Andernfalls geben Sie Ihren AWS geheimen Zugangsschlüssel ein.

Item	Description
Standardmäßiger Regionsname	Drücken Sie die Eingabetaste. Die Standardregion ist us-east-2. Sie können dies auf us-east-1, us-west-1, us-west-2, ap-south-1, ap-northeast-2, ap-southeast-1, ap-southeast-2, ap-northeast-1, eu-central-1, eu-west-1, oder sa-east-1 einstellen.
Standard-Ausgabeformat	Lassen Sie das Feld leer, und drücken Sie die Eingabetaste.
Pfad der hochzuladenden Protokolldatei	Der Speicherort der Datei, der die Protokolldaten enthält, die Sie senden möchten. Das Installationsprogramm schlägt Ihnen einen Pfad vor.
Ziel-Protokollgruppenname	Der Name für Ihre Protokollgruppe. Das Installationsprogramm schlägt Ihnen einen Protokollgruppenamen vor.
Ziel-Protokoll-Streamname	Standardmäßig ist dies der Name des Hosts. Das Installationsprogramm schlägt Ihnen einen Hostnamen vor.
Zeitstempelformat	Geben Sie das Format des Zeitstempels innerhalb der angegebenen Protokolldatei an. Wählen Sie „benutzerdefiniert“, um ihr eigenes Format anzugeben.
Ausgangsposition	Wie die Daten hochgeladen werden. Legen Sie diesen Wert auf start_of_file fest, um alles in die Datendatei hochzuladen. Legen Sie den Wert auf end_of_file fest, um nur neu hinzugefügte Daten hochzuladen.

Nachdem Sie diese Schritte ausgeführt haben, fragt das Installationsprogramm Sie, ob Sie eine weitere Protokolldatei konfigurieren möchten. Sie können den Vorgang für jede Protokolldatei beliebig oft durchführen. Wenn keine weiteren Protokolldateien überwacht werden sollen und ein weiteres Protokoll eingerichtet werden soll, wählen Sie N, wenn Sie vom Installationsprogramm dazu aufgefordert werden. Weitere Informationen zu den Einstellungen in der Konfigurationsdatei des Agenten finden Sie unter [CloudWatch Referenz zum Logs-Agenten](#).

Note

Die Konfiguration von mehreren Protokollquellen, damit Daten an einen einzelnen Protokoll-Stream gesendet werden, wird nicht unterstützt.

3. Ihnen sollten in der CloudWatch-Konsole einen kurzen Moment nach dem Ausführen des Agenten die neu erstellte Protokollgruppe und der Protokoll-Stream angezeigt werden.

Weitere Informationen finden Sie unter [An Logs gesendete Protokolldaten anzeigen CloudWatch](#).

Schnellstart: Installieren und konfigurieren Sie den CloudWatch Logs-Agent beim Start auf einer EC2-Linux-Instance

Tip

Der ältere CloudWatch Logs-Agent, der in diesem Abschnitt beschrieben wird, ist inzwischen veraltet. Wir empfehlen dringend, stattdessen den neuen vereinheitlichten CloudWatch Agenten zu verwenden, der sowohl Logs als auch Metriken erfassen kann. Darüber hinaus benötigt der ältere CloudWatch Logs-Agent Python 3.3 oder früher, und diese Versionen sind standardmäßig nicht auf neuen EC2-Instances installiert. Weitere Informationen zum Unified CloudWatch Agent finden Sie unter [Installation des CloudWatch Agenten](#).

Im Rest dieses Abschnitts wird die Verwendung des älteren CloudWatch Logs-Agenten erklärt.

Installation des älteren CloudWatch Logs-Agenten auf einer EC2-Linux-Instance beim Start

Sie können Amazon EC2-Benutzerdaten verwenden, eine Funktion von Amazon EC2, mit der parametrische Informationen beim Start an die Instance übergeben werden können, um den CloudWatch Logs-Agenten auf dieser Instance zu installieren und zu konfigurieren. Um die Installations- und Konfigurationsinformationen des CloudWatch Logs-Agenten an Amazon EC2 weiterzuleiten, können Sie die Konfigurationsdatei an einem Netzwerkspeicherort wie einem Amazon S3 S3-Bucket bereitstellen.

Die Konfiguration von mehreren Protokollquellen, damit Daten an einen einzelnen Protokoll-Stream gesendet werden, wird nicht unterstützt.

Voraussetzung

Erstellen Sie eine Konfigurationsdatei für den Agenten, in der alle Ihre Protokollgruppen und Protokollstreams beschrieben werden. Dies ist eine Textdatei, in der die zu überwachenden Protokolldateien sowie die Protokollgruppen und die Protokollstreams, in die diese hochgeladen werden sollen, beschrieben werden. Der Agent nutzt diese Konfigurationsdatei und startet das Überwachen und Hochladen aller darin beschriebenen Protokolldateien. Weitere Informationen zu den Einstellungen in der Konfigurationsdatei des Agenten finden Sie unter [CloudWatch Referenz zum Logs-Agenten](#).

Im Folgenden finden Sie eine Beispiel-Agent-Konfigurationsdatei für Amazon Linux 2

```
[general]
state_file = /var/lib/awslogs/state/agent-state

[/var/log/messages]
file = /var/log/messages
log_group_name = /var/log/messages
log_stream_name = {instance_id}
datetime_format = %b %d %H:%M:%S
```

Im Folgenden finden Sie ein Beispiel-Agenten-Konfigurationsdatei für Ubuntu

```
[general]
state_file = /var/awslogs/state/agent-state

[/var/log/syslog]
file = /var/log/syslog
log_group_name = /var/log/syslog
log_stream_name = {instance_id}
datetime_format = %b %d %H:%M:%S
```

So konfigurieren Sie Ihre IAM-Rolle

1. Öffnen Sie unter <https://console.aws.amazon.com/iam/> die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Policies und Create Policy aus.

3. Wählen Sie auf der Seite Create Policy für Create Your Own Policy die Option Select aus. Weitere Informationen zur Erstellung benutzerdefinierter Richtlinien finden Sie unter [IAM-Richtlinien für Amazon EC2 im Amazon EC2](#) EC2-Benutzerhandbuch.
4. Geben Sie auf der Seite Review Policy im Feld Policy Name einen Namen für die Richtlinie ein.
5. Fügen Sie die folgende Richtlinie unter Policy Document ein:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

6. Wählen Sie Richtlinie erstellen aus.
7. Wählen Sie im Navigationsbereich Roles und Create New Role aus.
8. Geben Sie auf der Seite Set Role Name einen Namen für die Rolle ein und wählen Sie dann Next Step aus.
9. Wählen Sie auf der Seite Select Role Type neben Amazon EC2 die Option Select aus.

10. Wählen Sie auf der Seite Attach Policy im Tabellenkopf die Option Policy Type und Customer Managed aus.
11. Wählen Sie die von Ihnen erstellte IAM-Richtlinie aus und anschließend Next Step (Nächster Schritt).
12. Wählen Sie Rolle erstellen aus.

Weitere Informationen zu Benutzern und Richtlinien finden Sie unter [IAM-Benutzer und -Gruppen](#) sowie unter [Verwalten von IAM-Richtlinien](#) im IAM-Benutzerhandbuch.

Um eine neue Instance zu starten und Logs zu aktivieren CloudWatch

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Wählen Sie Launch Instance aus.

Weitere Informationen finden Sie im Amazon EC2 EC2-Benutzerhandbuch unter [Launching an Instance](#).

3. Wählen Sie auf der Seite Step 1: Choose an Amazon Machine Image (AMI) den Linux-Instance-Typ zum Starten aus und wählen Sie anschließend auf der Seite Step 2: Choose an Instance Type die Option Next: Configure Instance Details.

Stellen Sie sicher, dass [cloud-init](#) in Ihr Amazon Machine Image (AMI) eingebunden ist. Amazon Linux AMIs und AMIs für Ubuntu und RHEL enthalten Cloud-Init bereits, CentOS und andere AMIs jedoch möglicherweise nicht. AWS Marketplace

4. Wählen Sie auf der Seite Schritt 3 Konfigurieren von Instance-Details für IAM-Rolle die IAM-Rolle, die Sie erstellt haben.
5. Fügen Sie unter Advanced Details für User data das folgende Skript in das Feld ein. Aktualisieren Sie dann das Skript, indem Sie den Wert der Option -c in den Speicherort Ihrer Agenten-Konfigurationsdatei ändern:

```
#!/bin/bash
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py -O
chmod +x ./awslogs-agent-setup.py
./awslogs-agent-setup.py -n -r us-east-1 -c s3://amzn-s3-demo-bucket/my-config-file
```

6. Nehmen Sie weitere Änderungen an der Instance vor, prüfen Sie die Starteinstellungen und wählen Sie dann Launch aus.

7. Sie sollten die neu erstellte Protokollgruppe und den Protokollstream in der CloudWatch Konsole sehen, nachdem der Agent einige Zeit lang ausgeführt wurde.

Weitere Informationen finden Sie unter [An Logs gesendete Protokolldaten anzeigen CloudWatch](#).

Schnellstart: Ermöglichen Sie Ihren Amazon EC2 EC2-Instances, auf denen Windows Server 2016 ausgeführt wird, mithilfe des Logs-Agenten CloudWatch Protokolle an CloudWatch Logs zu senden

 Tip

CloudWatch beinhaltet einen neuen vereinheitlichten Agenten, der sowohl Protokolle als auch Metriken von EC2-Instances und lokalen Servern sammeln kann. Wir empfehlen die Verwendung des neueren vereinheitlichten CloudWatch-Agents. Weitere Informationen finden Sie unter [Erste Schritte mit CloudWatch Logs](#).

Im Rest dieses Abschnitts wird die Verwendung des älteren CloudWatch Logs-Agenten erklärt.

Aktivieren Sie Ihre Amazon EC2 EC2-Instances, auf denen Windows Server 2016 ausgeführt wird, mithilfe des älteren CloudWatch Logs-Agenten zum Senden von CloudWatch Protokollen an Logs

Es gibt mehrere Methoden, mit denen Sie Instanzen, auf denen Windows Server 2016 ausgeführt wird, das Senden von Protokollen an CloudWatch Logs ermöglichen können. Die Schritte in diesem Abschnitt verwenden den Systems-Manager-Run-Command. Informationen zu den anderen möglichen Methoden finden Sie unter [Protokolle, Ereignisse und Leistungsindikatoren an Amazon CloudWatch senden](#).

Schritte

- [Herunterladen der Beispiel-Konfigurationsdatei](#)
- [Konfigurieren Sie die JSON-Datei für CloudWatch](#)
- [Erstellen einer IAM-Rolle für Systems Manager](#)
- [Überprüfen der Voraussetzungen für Systems Manager](#)
- [Überprüfen des Internetzugangs](#)

- [Aktivieren von CloudWatch Protokollen mithilfe des Systems Manager Manager-Befehls „Ausführen“](#)

Herunterladen der Beispiel-Konfigurationsdatei

Laden Sie die folgenden Beispiel-Datei auf Ihren Computer herunter:

[AWS.EC2.Windows.CloudWatch.json](#).

Konfigurieren Sie die JSON-Datei für CloudWatch

Sie bestimmen, an welche Protokolle gesendet CloudWatch werden sollen, indem Sie Ihre Optionen in einer Konfigurationsdatei angeben. Das Erstellen dieser Datei und die Angabe Ihrer Auswahl können 30 Minuten oder mehr beanspruchen. Nachdem Sie diese Aufgabe abgeschlossen haben, können Sie die Konfigurationsdatei für alle Instances wiederverwenden.

Schritte

- [Schritt 1: CloudWatch Logs aktivieren](#)
- [Schritt 2: Konfigurieren Sie die Einstellungen für CloudWatch](#)
- [Schritt 3: Konfigurieren der zu sendenden Daten](#)
- [Schritt 4: Konfigurieren der Ablaufsteuerung](#)
- [Schritt 5: Speichern von JSON-Inhalten](#)

Schritt 1: CloudWatch Logs aktivieren

Ändern Sie oben in der JSON-Datei "false" in "true" für `IsEnabled`:

```
"IsEnabled": true,
```

Schritt 2: Konfigurieren Sie die Einstellungen für CloudWatch

Geben Sie die Anmeldeinformationen, die Region, eine Protokollgruppennamen und einen Protokoll-Stream-Namespace ein. Dadurch kann die Instanz Protokolldaten an CloudWatch Logs senden. Um dieselben Protokolldaten an verschiedene Speicherorte zu senden, können Sie zusätzliche Abschnitte mit eindeutigen IDs (z. B. „CloudWatchLogs2“ und CloudWatchLogs 3“) und einer anderen Region für jede ID hinzufügen.

Um Einstellungen für das Senden von Protokolldaten an CloudWatch Logs zu konfigurieren

1. Suchen Sie in der JSON-Datei nach dem Abschnitt `CloudWatchLogs`.

```
{
  "Id": "CloudWatchLogs",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "AccessKey": "",
    "SecretKey": "",
    "Region": "us-east-1",
    "LogGroup": "Default-Log-Group",
    "LogStream": "{instance_id}"
  }
},
```

2. Lassen Sie die Felder `AccessKey` und `SecretKey` leer. Sie können Anmeldeinformationen mithilfe einer IAM-Rolle konfigurieren.
3. Geben Sie für `Region` die Region ein, an die Sie die Protokolldaten senden möchten (z. B. `us-east-2`).
4. Geben Sie in das Feld `LogGroup` den Namen Ihrer Protokollgruppe ein. Dieser Name wird auf dem Bildschirm Log Groups in der CloudWatch-Konsole angezeigt.
5. Geben Sie für `LogStream` den Ziel-Protokoll-Stream ein. Dieser Name wird auf dem Bildschirm Protokollgruppen > Streams in der CloudWatch Konsole angezeigt.

Wenn Sie `{instance_id}` verwenden, ist der Standard-Protokoll-Stream die Instance-ID dieser Instance.

Wenn Sie einen Log-Stream-Namen angeben, der noch nicht existiert, erstellt CloudWatch Logs ihn automatisch für Sie. Sie können den Protokoll-Stream-Namen mithilfe einer Literalzeichenfolge oder den vordefinierten Variablen `{instance_id}`, `{hostname}` und `{ip_address}` oder einer Kombination aus beiden definieren.

Schritt 3: Konfigurieren der zu sendenden Daten

Sie können Ereignisprotokolldaten, ETW-Daten (Event Tracing for Windows) und andere Protokolldaten an CloudWatch Logs senden.

Um Ereignisprotokolldaten von Windows-Anwendungen an Logs zu senden CloudWatch

1. Suchen Sie in der JSON-Datei nach dem Abschnitt `ApplicationEventLog`.

```
{
  "Id": "ApplicationEventLog",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "LogName": "Application",
    "Levels": "1"
  }
},
```

2. Geben Sie für `Levels` den hochzuladenden Meldungstyp an. Sie können einen der folgenden Werte angeben:

- **1** - Nur Fehlermeldungen hochladen.
- **2** - Nur Warnmeldungen hochladen.
- **4** - Nur Informationsmeldungen hochladen.

Sie können Werte kombinieren, um mehr als einen Meldungstyp einzuschließen. Beispielsweise lädt ein Wert von **3** Fehlermeldungen (**1**) und Warnmeldungen (**2**) hoch. Ein Wert von **7** lädt Fehlermeldungen (**1**), Warnmeldungen (**2**) und Informationsmeldungen (**4**) hoch.

Um Sicherheitsprotokolldaten an CloudWatch Logs zu senden

1. Suchen Sie in der JSON-Datei nach dem Abschnitt `SecurityEventLog`.

```
{
  "Id": "SecurityEventLog",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "LogName": "Security",
    "Levels": "7"
  }
},
```

2. Geben Sie für `Levels` **7** ein, um alle Meldungen hochzuladen.

Um Daten aus dem Systemereignisprotokoll an CloudWatch Logs zu senden

1. Suchen Sie in der JSON-Datei nach dem Abschnitt `SystemEventLog`.

```
{
  "Id": "SystemEventLog",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "LogName": "System",
    "Levels": "7"
  }
},
```

2. Geben Sie für `Levels` den hochzuladenden Meldungstyp an. Sie können einen der folgenden Werte angeben:

- **1** - Nur Fehlermeldungen hochladen.
- **2** - Nur Warnmeldungen hochladen.
- **4** - Nur Informationsmeldungen hochladen.

Sie können Werte kombinieren, um mehr als einen Meldungstyp einzuschließen. Beispielsweise lädt ein Wert von **3** Fehlermeldungen (**1**) und Warnmeldungen (**2**) hoch. Ein Wert von **7** lädt Fehlermeldungen (**1**), Warnmeldungen (**2**) und Informationsmeldungen (**4**) hoch.

Um andere Arten von Ereignisprotokolldaten an CloudWatch Logs zu senden

1. Fügen Sie der JSON-Datei einen neuen Abschnitt hinzu. Jeder Abschnitt muss eine eindeutige `Id` haben.

```
{
  "Id": "Id-name",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "LogName": "Log-name",
    "Levels": "7"
  }
},
```

2. Geben Sie für `Id` einen Namen für das hochzuladende Protokoll ein (z. B. **WindowsBackup**).
3. Geben Sie für `LogName` den Namen des hochzuladenden Protokolls ein. Sie können den Namen des Protokolls wie folgt suchen.
 - a. Öffnen Sie die Ereignisanzeige.
 - b. Wählen Sie im Navigationsbereich Applications and Services Logs aus.
 - c. Navigieren Sie zum Protokoll, und wählen Sie dann Actions und Properties aus.
4. Geben Sie für `Levels` den hochzuladenden Meldungstyp an. Sie können einen der folgenden Werte angeben:
 - **1** - Nur Fehlermeldungen hochladen.
 - **2** - Nur Warnmeldungen hochladen.
 - **4** - Nur Informationsmeldungen hochladen.

Sie können Werte kombinieren, um mehr als einen Meldungstyp einzuschließen. Beispielsweise lädt ein Wert von **3** Fehlermeldungen (**1**) und Warnmeldungen (**2**) hoch. Ein Wert von **7** lädt Fehlermeldungen (**1**), Warnmeldungen (**2**) und Informationsmeldungen (**4**) hoch.

Um Daten der Ereignisablaufverfolgung für Windows an Logs zu CloudWatch senden

ETW (Event Tracing for Windows) bietet einen effizienten und detaillierten Protokollierungsmechanismus, auf den Anwendungen Protokolle schreiben können. Jeder ETW wird über einen Session Manager gesteuert, der die Protokollierungssitzung starten und beenden kann. Jede Sitzung hat einen Anbieter und einen oder mehrere Verbraucher.

1. Suchen Sie in der JSON-Datei nach dem Abschnitt ETW.

```
{
  "Id": "ETW",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "LogName": "Microsoft-Windows-WinINet/Analytic",
    "Levels": "7"
  }
},
```

2. Geben Sie für `LogName` den Namen des hochzuladenden Protokolls ein.

3. Geben Sie für `Levels` den hochzuladenden Meldungstyp an. Sie können einen der folgenden Werte angeben:

- **1** - Nur Fehlermeldungen hochladen.
- **2** - Nur Warnmeldungen hochladen.
- **4** - Nur Informationsmeldungen hochladen.

Sie können Werte kombinieren, um mehr als einen Meldungstyp einzuschließen. Beispielsweise lädt ein Wert von **3** Fehlermeldungen (**1**) und Warnmeldungen (**2**) hoch. Ein Wert von **7** lädt Fehlermeldungen (**1**), Warnmeldungen (**2**) und Informationsmeldungen (**4**) hoch.

Um benutzerdefinierte Protokolle (jede textbasierte Protokolldatei) an Logs zu senden CloudWatch

1. Suchen Sie in der JSON-Datei nach dem Abschnitt `CustomLogs`.

```
{
  "Id": "CustomLogs",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "LogDirectoryPath": "C:\\\\CustomLogs\\\\",
    "TimestampFormat": "MM/dd/yyyy HH:mm:ss",
    "Encoding": "UTF-8",
    "Filter": "",
    "CultureName": "en-US",
    "TimeZoneKind": "Local",
    "LineCount": "5"
  }
},
```

2. Geben Sie für `LogDirectoryPath` den Pfad ein, auf dem die Protokolle auf Ihrer Instance gespeichert werden sollen.
3. Geben Sie unter `TimestampFormat` das Zeitstempelformat ein, das Sie verwenden möchten. Weitere Informationen über unterstützte Werte finden Sie unter dem Thema [Benutzerdefinierte Datums- und Zeitformat-Zeichenfolgen](#) auf MSDN.

 Important

Die Quell-Protokolldatei muss zu Beginn jeder Protokollzeile einen Zeitstempel haben, und nach dem Zeitstempel muss eine Leerstelle folgen.

4. Geben Sie für `Encoding` die zu verwendende Datei-Kodierung ein (z. B: UTF-8). Eine Liste der unterstützten Werte finden Sie unter dem Thema [Encoding Class](#) auf MSDN.

 Note

Verwenden Sie den Kodierungsnamen, nicht den Anzeigenamen.

5. (Optional) Geben Sie für `Filter` das Präfix des Protokollnamens ein. Lassen Sie diesen Parameter leer, um alle Dateien zu überwachen. Weitere Informationen zu unterstützten Werten finden Sie unter dem Thema [FileSystemWatcherFilter Eigenschaften](#) auf MSDN.
6. (Optional) Geben Sie für `CultureName` das Gebietsschema ein, unter dem Zeitstempel protokolliert wird. Wenn `CultureName` leer ist, wird standardmäßig dasselbe Gebietsschema verwendet, das von der Windows-Instance verwendet wird. Weitere Informationen finden Sie in der Spalte `Language tag` in der Tabelle im Thema [Produktverhalten](#) in MSDN.

 Note

Die Werte `div`, `div-MV`, `hu` und `hu-HU` werden nicht unterstützt.

7. (Optional) Geben Sie für `TimeZoneKind` `Local` oder `UTC` ein. Sie können über diese Einstellung Zeitzeoneninformationen bereitstellen, wenn der Zeitstempel Ihres Protokolls keine Zeitzeoneninformationen enthält. Wenn dieser Parameter leer gelassen wird und Ihr Zeitstempel keine Zeitzeoneninformationen enthält, verwendet CloudWatch Logs standardmäßig die lokale Zeitzone. Dieser Parameter wird ignoriert, wenn der Zeitstempel bereits Zeitzeoneninformationen enthält.
8. (Optional) Geben Sie für `LineCount` die Anzahl der Zeilen im Header ein, um die Protokolldatei zu identifizieren. Beispielsweise haben IIS-Protokolldateien praktisch identische Header. Sie können `5` eingeben, dann würden die ersten drei Zeilen des Headers der Protokolldatei gelesen, um diese zu identifizieren. In den IIS-Protokolldateien ist die dritte Zeile das Datum und der Zeitstempel, aber es ist nicht immer sichergestellt, dass der Zeitstempel von zwei verschiedenen

Protokolldateien unterschiedlich ist. Aus diesem Grund empfehlen wir, mindestens eine Zeile der tatsächlichen Protokolldaten hinzuzufügen, so dass die Protokolldatei eindeutig identifizierbar ist.

Um IIS-Protokolldaten an Logs zu CloudWatch senden

1. Suchen Sie in der JSON-Datei nach dem Abschnitt IISLog.

```
{
  "Id": "IISLogs",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "LogDirectoryPath": "C:\\inetpub\\logs\\LogFiles\\W3SVC1",
    "TimestampFormat": "yyyy-MM-dd HH:mm:ss",
    "Encoding": "UTF-8",
    "Filter": "",
    "CultureName": "en-US",
    "TimeZoneKind": "UTC",
    "LineCount": "5"
  }
},
```

2. Geben Sie für LogDirectoryPath das Verzeichnis an, in dem die IIS-Protokolldateien für einen individuellen Standort gespeichert sind (z. B. C:\inetpub\logs\LogFiles\W3SVC1).

 Note

Es wird nur das W3C-Protokollformat unterstützt. IIS, NCSA und benutzerdefinierte Formate werden nicht unterstützt.

3. Geben Sie unter TimestampFormat das Zeitstempelformat ein, das Sie verwenden möchten. Weitere Informationen über unterstützte Werte finden Sie unter dem Thema [Benutzerdefinierte Datums- und Zeitformat-Zeichenfolgen](#) auf MSDN.
4. Geben Sie für Encoding die zu verwendende Datei-Kodierung ein (z. B. UTF-8). Weitere Informationen über unterstützte Werte finden Sie unter dem Thema [Encoding Class](#) auf MSDN.

 Note

Verwenden Sie den Kodierungsnamen, nicht den Anzeigenamen.

5. (Optional) Geben Sie für `Filter` das Präfix des Protokollnamens ein. Lassen Sie diesen Parameter leer, um alle Dateien zu überwachen. Weitere Informationen zu unterstützten Werten finden Sie unter dem Thema [FileSystemWatcherFilter Eigenschaften](#) auf MSDN.
6. (Optional) Geben Sie für `CultureName` das Gebietsschema ein, unter dem Zeitstempel protokolliert wird. Wenn `CultureName` leer ist, wird standardmäßig dasselbe Gebietsschema verwendet, das von der Windows-Instance verwendet wird. Weitere Informationen über unterstützte Werte finden Sie in der Spalte `Language` tag in der Tabelle im Thema [Produktverhalten](#) in MSDN.

 Note

Die Werte `div`, `div-MV`, `hu` und `hu-HU` werden nicht unterstützt.

7. (Optional) Geben Sie für `TimeZoneKind` `Local` oder `UTC` ein. Sie können über diese Einstellung Zeitzoneinformationen bereitstellen, wenn der Zeitstempel Ihres Protokolls keine Zeitzoneinformationen enthält. Wenn dieser Parameter leer gelassen wird und Ihr Zeitstempel keine Zeitzoneinformationen enthält, verwendet CloudWatch Logs standardmäßig die lokale Zeitzone. Dieser Parameter wird ignoriert, wenn der Zeitstempel bereits Zeitzoneinformationen enthält.
8. (Optional) Geben Sie für `LineCount` die Anzahl der Zeilen im Header ein, um die Protokolldatei zu identifizieren. Beispielsweise haben IIS-Protokolldateien praktisch identische Header. Sie können `5` eingeben, dann würden die ersten fünf Zeilen des Headers der Protokolldatei gelesen, um diese zu identifizieren. In den IIS-Protokolldateien ist die dritte Zeile das Datum und der Zeitstempel, aber es ist nicht immer sichergestellt, dass der Zeitstempel von zwei verschiedenen Protokolldateien unterschiedlich ist. Aus diesem Grund empfehlen wir, mindestens eine Zeile der tatsächlichen Protokolldaten hinzuzufügen, sodass die Protokolldatei eindeutig identifizierbar ist.

Schritt 4: Konfigurieren der Ablaufsteuerung

Jeder Datentyp muss ein entsprechendes Ziel im Bereich `Flows` haben. Um beispielsweise das benutzerdefinierte Protokoll, das ETW-Protokoll und das Systemprotokoll an CloudWatch Logs zu senden, fügen Sie dem `Flows` Abschnitt etwas (`CustomLogs`, `ETW`, `SystemEventLog`), `CloudWatchLogs` hinzu.

⚠ Warning

Wenn Sie einen ungültigen Schritt hinzufügen, ist der Fluss blockiert. Wenn Sie beispielsweise einen Metrikschritt für eine Festplatte hinzufügen, Ihre Instance aber keine Festplatte hat, sind alle Schritte im Fluss blockiert.

Sie können dieselbe Protokolldatei an mehrere Ziele senden. Wenn Sie beispielsweise das Anwendungsprotokoll an zwei unterschiedliche Ziele senden möchten, die Sie im Abschnitt CloudWatchLogs definiert haben, fügen Sie dem Abschnitt Flows Folgendes hinzu: `ApplicationEventLog, (CloudWatchLogs, CloudWatchLogs2)`.

So konfigurieren Sie die Flussteuerung:

1. Suchen Sie in der Datei `AWS.EC2.Windows.CloudWatch.json` den Abschnitt Flows.

```
"Flows": {
  "Flows": [
    "PerformanceCounter,CloudWatch",
    "(PerformanceCounter,PerformanceCounter2), CloudWatch2",
    "(CustomLogs, ETW, SystemEventLog),CloudWatchLogs",
    "CustomLogs, CloudWatchLogs2",
    "ApplicationEventLog,(CloudWatchLogs, CloudWatchLogs2)"
  ]
}
```

2. Geben Sie für Flows jeden Datentyp ein, der hochgeladen werden soll (z. B. `ApplicationEventLog`) sowie sein Ziel (z. B. `CloudWatchLogs`).

Schritt 5: Speichern von JSON-Inhalten

Sie haben jetzt die Bearbeitung der JSON-Datei abgeschlossen. Speichern Sie sie und fügen Sie den Dateiinhalt in einem anderen Fenster in einen Texteditor ein. Sie benötigen den Dateiinhalt in einem späteren Schritt dieses Vorgangs.

Erstellen einer IAM-Rolle für Systems Manager

Sie benötigen eine IAM-Rolle für Instance-Anmeldeinformationen, wenn Sie den Systems-Manager-Run-Command verwenden. Diese Rolle ermöglicht es Systems Manager, Aktionen auf der Instance auszuführen. Weitere Informationen finden Sie unter [Konfigurieren von Sicherheitsrollen für Systems](#)

[Manager](#) im AWS Systems Manager -Benutzerhandbuch. Informationen zum Anhängen einer IAM-Rolle an eine bestehende Instance finden Sie unter [Anhängen einer IAM-Rolle an eine Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

Überprüfen der Voraussetzungen für Systems Manager

Bevor Sie Systems Manager Run Command verwenden, um die Integration mit CloudWatch Logs zu konfigurieren, stellen Sie sicher, dass Ihre Instances die Mindestanforderungen erfüllen. Weitere Informationen finden Sie unter [Voraussetzungen für Systems Manager](#) im AWS Systems Manager -Benutzerhandbuch.

Überprüfen des Internetzugangs

Ihre Amazon EC2 Windows Server-Instances und verwalteten Instances müssen über ausgehenden Internetzugang verfügen, um Protokoll- und Ereignisdaten an senden zu können. CloudWatch Weitere Informationen dazu, wie Sie den Internetzugang konfigurieren, finden Sie unter [Internet-Gateways](#) im Benutzerhandbuch zu Amazon VPC.

Aktivieren von CloudWatch Protokollen mithilfe des Systems Manager Manager-Befehls „Ausführen“

Run Command ermöglicht Ihnen die bedarfsgerechte Verwaltung der Konfiguration Ihrer Instances. Sie geben ein Systems-Manager-Dokument und Parameter an und führen Sie den Befehl auf einer oder mehreren Instances aus. Der SSM-Agent auf der Instance verarbeitet den Befehl und konfiguriert die Instance wie angegeben.

Um die Integration mit CloudWatch Logs mithilfe von Run Command zu konfigurieren

1. Öffnen Sie die Amazon-EC2-Konsole unter <https://console.aws.amazon.com/ec2/>.
2. Öffnen Sie die SSM-Konsole unter <https://console.aws.amazon.com/systems-manager/>.
3. Wählen Sie im Navigationsbereich Run Command aus.
4. Wählen Sie die Option Run a command.
5. Wählen Sie für Command document die Option AWS- ConfigureCloudWatch.
6. Wählen Sie für Target-Instances die Instances aus, die in CloudWatch Logs integriert werden sollen. Wenn keine Instance in der Liste angezeigt wird, ist möglicherweise für Run Command keine konfiguriert. Weitere Informationen finden Sie unter [Voraussetzungen für Systems Manager](#) im Amazon EC2 EC2-Benutzerhandbuch.
7. Wählen Sie für Status die Option Enabled.
8. Für Properties kopieren Sie die JSON-Inhalte, die Sie in den vorherigen Schritten erstellt haben, und fügen Sie diese ein.

9. Füllen Sie die verbleibenden optionalen Felder aus, und wählen Sie Run.

Verwenden Sie die folgende Vorgehensweise, um die Ergebnisse der Befehlsausführung in der Amazon-EC2-Konsole anzuzeigen.

So zeigen Sie die Befehlsausgabe in der Konsole an:

1. Wählen Sie einen Befehl.
2. Wählen Sie die Registerkarte Output aus.
3. Wählen Sie View Output aus. Der Befehlsausgabeseite zeigt die Ergebnisse der Befehlsausführung an.

Schnellstart: Aktivieren Sie Ihre Amazon EC2 EC2-Instances, auf denen Windows Server 2012 und Windows Server 2008 ausgeführt werden, um Protokolle an Logs zu CloudWatch senden

 Tip

CloudWatch enthält einen neuen vereinheitlichten Agenten, der sowohl Protokolle als auch Messwerte von EC2-Instances und lokalen Servern sammeln kann. Wir empfehlen die Verwendung des neueren vereinheitlichten CloudWatch-Agents. Weitere Informationen finden Sie unter [Erste Schritte mit CloudWatch Logs](#).

Im Rest dieses Abschnitts wird die Verwendung des älteren CloudWatch Logs-Agenten erklärt.

Aktivieren Sie Ihre Amazon EC2 EC2-Instances, auf denen Windows Server 2012 und Windows Server 2008 ausgeführt werden, um Protokolle an Logs zu CloudWatch senden

Gehen Sie wie folgt vor, damit Ihre Instanzen, auf denen Windows Server 2012 und Windows Server 2008 ausgeführt werden, CloudWatch Protokolle an Logs senden können.

Herunterladen der Beispiel-Konfigurationsdatei

Laden Sie die folgende Beispiel-JSON-Datei auf Ihren Computer herunter:

[AWS.EC2.Windows.CloudWatch.json](#). Sie werden sie in den folgenden Schritten bearbeiten.

Konfigurieren Sie die JSON-Datei für CloudWatch

Sie bestimmen, an welche Protokolle gesendet CloudWatch werden sollen, indem Sie Ihre Optionen in der JSON-Konfigurationsdatei angeben. Das Erstellen dieser Datei und die Angabe Ihrer Auswahl können 30 Minuten oder mehr beanspruchen. Nachdem Sie diese Aufgabe abgeschlossen haben, können Sie die Konfigurationsdatei für alle Instances wiederverwenden.

Schritte

- [Schritt 1: CloudWatch Logs aktivieren](#)
- [Schritt 2: Konfigurieren Sie die Einstellungen für CloudWatch](#)
- [Schritt 3: Konfigurieren der zu sendenden Daten](#)
- [Schritt 4: Konfigurieren der Ablaufsteuerung](#)

Schritt 1: CloudWatch Logs aktivieren

Ändern Sie oben in der JSON-Datei "false" in "true" für IsEnabled:

```
"IsEnabled": true,
```

Schritt 2: Konfigurieren Sie die Einstellungen für CloudWatch

Geben Sie die Anmeldeinformationen, die Region, eine Protokollgruppennamen und einen Protokoll-Stream-Namespace ein. Dadurch kann die Instanz Protokolldaten an CloudWatch Logs senden. Um dieselben Protokolldaten an verschiedene Speicherorte zu senden, können Sie zusätzliche Abschnitte mit eindeutigen IDs (z. B. „CloudWatchLogs2“ und CloudWatchLogs 3“) und einer anderen Region für jede ID hinzufügen.

Um Einstellungen für das Senden von Protokolldaten an CloudWatch Logs zu konfigurieren

1. Suchen Sie in der JSON-Datei nach dem Abschnitt CloudWatchLogs.

```
{
  "Id": "CloudWatchLogs",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "AccessKey": "",
    "SecretKey": "",
```

```

        "Region": "us-east-1",
        "LogGroup": "Default-Log-Group",
        "LogStream": "{instance_id}"
    },
},

```

2. Lassen Sie die Felder AccessKey und SecretKey leer. Sie können Anmeldeinformationen mithilfe einer IAM-Rolle konfigurieren.
3. Geben Sie für Region die Region ein, an die Sie die Protokolldaten senden möchten (z. B. us-east-2).
4. Geben Sie in das Feld LogGroup den Namen Ihrer Protokollgruppe ein. Dieser Name wird auf dem Bildschirm Log Groups in der CloudWatch-Konsole angezeigt.
5. Geben Sie für LogStream den Ziel-Protokoll-Stream ein. Dieser Name wird auf dem Bildschirm Protokollgruppen > Streams in der CloudWatch Konsole angezeigt.

Wenn Sie {instance_id} verwenden, ist der Standard-Protokoll-Stream die Instance-ID dieser Instance.

Wenn Sie einen Log-Stream-Namen angeben, der noch nicht existiert, erstellt CloudWatch Logs ihn automatisch für Sie. Sie können den Protokoll-Stream-Namen mithilfe einer Literalzeichenfolge oder den vordefinierten Variablen {instance_id}, {hostname} und {ip_address} oder einer Kombination aus beiden definieren.

Schritt 3: Konfigurieren der zu sendenden Daten

Sie können Ereignisprotokolldaten, ETW-Daten (Event Tracing for Windows) und andere Protokolldaten an CloudWatch Logs senden.

Um Ereignisprotokolldaten von Windows-Anwendungen an Logs zu senden CloudWatch

1. Suchen Sie in der JSON-Datei nach dem Abschnitt ApplicationEventLog.

```

{
    "Id": "ApplicationEventLog",
    "FullName":
    "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Application",
        "Levels": "1"
    }
}

```

```
},
```

2. Geben Sie für **Levels** den hochzuladenden Meldungstyp an. Sie können einen der folgenden Werte angeben:

- **1** - Nur Fehlermeldungen hochladen.
- **2** - Nur Warnmeldungen hochladen.
- **4** - Nur Informationsmeldungen hochladen.

Sie können Werte kombinieren, um mehr als einen Meldungstyp einzuschließen. Beispielsweise lädt ein Wert von **3** Fehlermeldungen (**1**) und Warnmeldungen (**2**) hoch. Ein Wert von **7** lädt Fehlermeldungen (**1**), Warnmeldungen (**2**) und Informationsmeldungen (**4**) hoch.

Um Sicherheitsprotokolldaten an CloudWatch Logs zu senden

1. Suchen Sie in der JSON-Datei nach dem Abschnitt **SecurityEventLog**.

```
{
  "Id": "SecurityEventLog",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "LogName": "Security",
    "Levels": "7"
  }
},
```

2. Geben Sie für **Levels** **7** ein, um alle Meldungen hochzuladen.

Um Daten aus dem Systemereignisprotokoll an CloudWatch Logs zu senden

1. Suchen Sie in der JSON-Datei nach dem Abschnitt **SystemEventLog**.

```
{
  "Id": "SystemEventLog",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "LogName": "System",
```



```
    "Levels": "7"
  }
},
```

2. Geben Sie für `Levels` den hochzuladenden Meldungstyp an. Sie können einen der folgenden Werte angeben:

- **1** - Nur Fehlermeldungen hochladen.
- **2** - Nur Warnmeldungen hochladen.
- **4** - Nur Informationsmeldungen hochladen.

Sie können Werte kombinieren, um mehr als einen Meldungstyp einzuschließen. Beispielsweise lädt ein Wert von **3** Fehlermeldungen (**1**) und Warnmeldungen (**2**) hoch. Ein Wert von **7** lädt Fehlermeldungen (**1**), Warnmeldungen (**2**) und Informationsmeldungen (**4**) hoch.

Um andere Arten von Ereignisprotokolldaten an CloudWatch Logs zu senden

1. Fügen Sie der JSON-Datei einen neuen Abschnitt hinzu. Jeder Abschnitt muss eine eindeutige Id haben.

```
{
  "Id": "Id-name",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "LogName": "Log-name",
    "Levels": "7"
  }
},
```

2. Geben Sie für `Id` einen Namen für das hochzuladende Protokoll ein (z. B. **WindowsBackup**).
3. Geben Sie für `LogName` den Namen des hochzuladenden Protokolls ein. Sie können den Namen des Protokolls wie folgt suchen.
 - a. Öffnen Sie die Ereignisanzeige.
 - b. Wählen Sie im Navigationsbereich Applications and Services Logs aus.
 - c. Navigieren Sie zum Protokoll, und wählen Sie dann Actions und Properties aus.

4. Geben Sie für `Levels` den hochzuladenden Meldungstyp an. Sie können einen der folgenden Werte angeben:

- **1** - Nur Fehlermeldungen hochladen.
- **2** - Nur Warnmeldungen hochladen.
- **4** - Nur Informationsmeldungen hochladen.

Sie können Werte kombinieren, um mehr als einen Meldungstyp einzuschließen. Beispielsweise lädt ein Wert von **3** Fehlermeldungen (**1**) und Warnmeldungen (**2**) hoch. Ein Wert von **7** lädt Fehlermeldungen (**1**), Warnmeldungen (**2**) und Informationsmeldungen (**4**) hoch.

Um Daten der Ereignisablaufverfolgung für Windows an Logs zu CloudWatch senden

ETW (Event Tracing for Windows) bietet einen effizienten und detaillierten Protokollierungsmechanismus, auf den Anwendungen Protokolle schreiben können. Jeder ETW wird über einen Session Manager gesteuert, der die Protokollierungssitzung starten und beenden kann. Jede Sitzung hat einen Anbieter und einen oder mehrere Verbraucher.

1. Suchen Sie in der JSON-Datei nach dem Abschnitt ETW.

```
{
  "Id": "ETW",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "LogName": "Microsoft-Windows-WinINet/Analytic",
    "Levels": "7"
  }
},
```

2. Geben Sie für `LogName` den Namen des hochzuladenden Protokolls ein.

3. Geben Sie für `Levels` den hochzuladenden Meldungstyp an. Sie können einen der folgenden Werte angeben:

- **1** - Nur Fehlermeldungen hochladen.
- **2** - Nur Warnmeldungen hochladen.
- **4** - Nur Informationsmeldungen hochladen.

Sie können Werte kombinieren, um mehr als einen Meldungstyp einzuschließen. Beispielsweise lädt ein Wert von **3** Fehlermeldungen (**1**) und Warnmeldungen (**2**) hoch. Ein Wert von **7** lädt Fehlermeldungen (**1**), Warnmeldungen (**2**) und Informationsmeldungen (**4**) hoch.

Um benutzerdefinierte Protokolle (jede textbasierte Protokolldatei) an Logs zu senden CloudWatch

1. Suchen Sie in der JSON-Datei nach dem Abschnitt CustomLogs.

```
{
  "Id": "CustomLogs",
  "FullName":
  "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
  "Parameters": {
    "LogDirectoryPath": "C:\\\\CustomLogs\\\\",
    "TimestampFormat": "MM/dd/yyyy HH:mm:ss",
    "Encoding": "UTF-8",
    "Filter": "",
    "CultureName": "en-US",
    "TimeZoneKind": "Local",
    "LineCount": "5"
  }
},
```

2. Geben Sie für LogDirectoryPath den Pfad ein, auf dem die Protokolle auf Ihrer Instance gespeichert werden sollen.
3. Geben Sie unter TimestampFormat das Zeitstempelformat ein, das Sie verwenden möchten. Weitere Informationen über unterstützte Werte finden Sie unter dem Thema [Benutzerdefinierte Datums- und Zeitformat-Zeichenfolgen](#) auf MSDN.

 Important

Die Quell-Protokolldatei muss zu Beginn jeder Protokollzeile einen Zeitstempel haben, und nach dem Zeitstempel muss eine Leerstelle folgen.

4. Geben Sie für Encoding die zu verwendende Datei-Kodierung ein (z. B: UTF-8). Weitere Informationen über unterstützte Werte finden Sie unter dem Thema [Encoding Class](#) auf MSDN.

 Note

Verwenden Sie den Kodierungsnamen, nicht den Anzeigenamen.

5. (Optional) Geben Sie für `Filter` das Präfix des Protokollnamens ein. Lassen Sie diesen Parameter leer, um alle Dateien zu überwachen. Weitere Informationen zu unterstützten Werten finden Sie unter dem Thema [FileSystemWatcherFilter Eigenschaften](#) auf MSDN.
6. (Optional) Geben Sie für `CultureName` das Gebietsschema ein, unter dem Zeitstempel protokolliert wird. Wenn `CultureName` leer ist, wird standardmäßig dasselbe Gebietsschema verwendet, das von der Windows-Instance verwendet wird. Weitere Informationen über unterstützte Werte finden Sie in der Spalte `Language` tag in der Tabelle im Thema [Produktverhalten](#) in MSDN.

 Note

Die Werte `div`, `div-MV`, `hu` und `hu-HU` werden nicht unterstützt.

7. (Optional) Geben Sie für `TimeZoneKind` `Local` oder `UTC` ein. Sie können über diese Einstellung Zeitzoneinformationen bereitstellen, wenn der Zeitstempel Ihres Protokolls keine Zeitzoneinformationen enthält. Wenn dieser Parameter leer gelassen wird und Ihr Zeitstempel keine Zeitzoneinformationen enthält, verwendet CloudWatch Logs standardmäßig die lokale Zeitzone. Dieser Parameter wird ignoriert, wenn der Zeitstempel bereits Zeitzoneinformationen enthält.
8. (Optional) Geben Sie für `LineCount` die Anzahl der Zeilen im Header ein, um die Protokolldatei zu identifizieren. Beispielsweise haben IIS-Protokolldateien praktisch identische Header. Sie können `5` eingeben, dann würden die ersten drei Zeilen des Headers der Protokolldatei gelesen, um diese zu identifizieren. In den IIS-Protokolldateien ist die dritte Zeile das Datum und der Zeitstempel, aber es ist nicht immer sichergestellt, dass der Zeitstempel von zwei verschiedenen Protokolldateien unterschiedlich ist. Aus diesem Grund empfehlen wir, mindestens eine Zeile der tatsächlichen Protokolldaten hinzuzufügen, so dass die Protokolldatei eindeutig identifizierbar ist.

Um IIS-Protokolldaten an Logs zu CloudWatch senden

1. Suchen Sie in der JSON-Datei nach dem Abschnitt `IISLog`.

```
{
```

```
"Id": "IISLogs",
"FullName":
"AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
"Parameters": {
  "LogDirectoryPath": "C:\\inetpub\\logs\\LogFiles\\W3SVC1",
  "TimestampFormat": "yyyy-MM-dd HH:mm:ss",
  "Encoding": "UTF-8",
  "Filter": "",
  "CultureName": "en-US",
  "TimeZoneKind": "UTC",
  "LineCount": "5"
},
```

2. Geben Sie für `LogDirectoryPath` das Verzeichnis an, in dem die IIS-Protokolldateien für einen individuellen Standort gespeichert sind (z. B. `C:\inetpub\logs\LogFiles\W3SVC1`).

 Note

Es wird nur das W3C-Protokollformat unterstützt. IIS, NCSA und benutzerdefinierte Formate werden nicht unterstützt.

3. Geben Sie unter `TimestampFormat` das Zeitstempelformat ein, das Sie verwenden möchten. Weitere Informationen über unterstützte Werte finden Sie unter dem Thema [Benutzerdefinierte Datums- und Zeitformat-Zeichenfolgen](#) auf MSDN.
4. Geben Sie für `Encoding` die zu verwendende Datei-Kodierung ein (z. B. UTF-8). Weitere Informationen über unterstützte Werte finden Sie unter dem Thema [Encoding Class](#) auf MSDN.

 Note

Verwenden Sie den Kodierungsnamen, nicht den Anzeigenamen.

5. (Optional) Geben Sie für `Filter` das Präfix des Protokollnamens ein. Lassen Sie diesen Parameter leer, um alle Dateien zu überwachen. Weitere Informationen zu unterstützten Werten finden Sie unter dem Thema [FileSystemWatcherFilter Eigenschaften](#) auf MSDN.
6. (Optional) Geben Sie für `CultureName` das Gebietsschema ein, unter dem Zeitstempel protokolliert wird. Wenn `CultureName` leer ist, wird standardmäßig dasselbe Gebietsschema verwendet, das von der Windows-Instance verwendet wird. Weitere Informationen über

unterstützte Werte finden Sie in der Spalte `Language tag` in der Tabelle im Thema [Produktverhalten](#) in MSDN.

 Note

Die Werte `div`, `div-MV`, `hu` und `hu-HU` werden nicht unterstützt.

7. (Optional) Geben Sie für `TimeZoneKind` `Local` oder `UTC` ein. Sie können über diese Einstellung Zeitzoneneinformationen bereitstellen, wenn der Zeitstempel Ihres Protokolls keine Zeitzoneneinformationen enthält. Wenn dieser Parameter leer gelassen wird und Ihr Zeitstempel keine Zeitzoneneinformationen enthält, verwendet CloudWatch Logs standardmäßig die lokale Zeitzone. Dieser Parameter wird ignoriert, wenn der Zeitstempel bereits Zeitzoneneinformationen enthält.
8. (Optional) Geben Sie für `LineCount` die Anzahl der Zeilen im Header ein, um die Protokolldatei zu identifizieren. Beispielsweise haben IIS-Protokolldateien praktisch identische Header. Sie können `5` eingeben, dann würden die ersten fünf Zeilen des Headers der Protokolldatei gelesen, um diese zu identifizieren. In den IIS-Protokolldateien ist die dritte Zeile das Datum und der Zeitstempel, aber es ist nicht immer sichergestellt, dass der Zeitstempel von zwei verschiedenen Protokolldateien unterschiedlich ist. Aus diesem Grund empfehlen wir, mindestens eine Zeile der tatsächlichen Protokolldaten hinzuzufügen, sodass die Protokolldatei eindeutig identifizierbar ist.

Schritt 4: Konfigurieren der Ablaufsteuerung

Jeder Datentyp muss ein entsprechendes Ziel im Bereich `Flows` haben. Um beispielsweise das benutzerdefinierte Protokoll, das ETW-Protokoll und das Systemprotokoll an CloudWatch Logs zu senden, fügen Sie dem `Flows` Abschnitt etwas (`CustomLogs`, `ETW`, `SystemEventLog`), `CloudWatchLogs` hinzu.

 Warning

Wenn Sie einen ungültigen Schritt hinzufügen, ist der Fluss blockiert. Wenn Sie beispielsweise einen Metrikschritt für eine Festplatte hinzufügen, Ihre Instance aber keine Festplatte hat, sind alle Schritte im Fluss blockiert.

Sie können dieselbe Protokolldatei an mehrere Ziele senden. Wenn Sie beispielsweise das Anwendungsprotokoll an zwei unterschiedliche Ziele senden möchten, die Sie im Abschnitt

CloudWatchLogs definiert haben, fügen Sie dem Abschnitt Flows Folgendes hinzu: ApplicationEventLog, (CloudWatchLogs, CloudWatchLogs2).

So konfigurieren Sie die Flussteuerung:

1. Suchen Sie in der Datei AWS.EC2.Windows.CloudWatch.json den Abschnitt Flows.

```
"Flows": {
  "Flows": [
    "PerformanceCounter,CloudWatch",
    "(PerformanceCounter,PerformanceCounter2), CloudWatch2",
    "(CustomLogs, ETW, SystemEventLog),CloudWatchLogs",
    "CustomLogs, CloudWatchLogs2",
    "ApplicationEventLog,(CloudWatchLogs, CloudWatchLogs2)"
  ]
}
```

2. Geben Sie für Flows jeden Datentyp ein, der hochgeladen werden soll (z. B. ApplicationEventLog) sowie sein Ziel (z. B. CloudWatchLogs).

Sie haben jetzt die Bearbeitung der JSON-Datei abgeschlossen. Sie werden sie in einem späteren Schritt verwenden.

Den Agenten starten

Um einer Amazon EC2 EC2-Instance, auf der Windows Server 2012 oder Windows Server 2008 ausgeführt wird, das Senden von Protokollen an CloudWatch Logs zu ermöglichen, verwenden Sie den EC2 Config-Service (EC2Config.exe). Ihre Instanz sollte EC2 Config 4.0 oder höher haben, und Sie können dieses Verfahren verwenden.

So konfigurieren Sie CloudWatch mit EC2 Config 4.x

1. Prüfen Sie die Kodierung der Datei AWS.EC2.Windows.CloudWatch.json, die Sie zu einem früheren Zeitpunkt in dieser Anleitung bearbeitet haben. Es wird nur UTF-8 ohne BOM-Kodierung unterstützt. Speichern Sie die Datei in folgendem Verzeichnis auf Windows Server 2008 – 2012 R2-Instance: C:\Program Files\Amazon\SSM\Plugins\awsCloudWatch\.
2. Starten oder starten Sie den SSM-Agenten (AmazonSSMAgent.exe) über die Systemsteuerung von Windows Services oder mit dem folgenden PowerShell Befehl neu:

```
PS C:\> Restart-Service AmazonSSMAgent
```

Nach dem Neustart erkennt der SSM-Agent die Konfigurationsdatei und konfiguriert die Instanz für die Integration. CloudWatch Wenn Sie die Parameter und Einstellungen in der lokalen Konfigurationsdatei ändern, müssen Sie den SSM-Agent neu starten, damit die Änderungen übernommen werden. Um die CloudWatch Integration auf der Instanz zu deaktivieren, wechseln Sie `IsEnabled` zur Konfigurationsdatei `false` und speichern Sie Ihre Änderungen in der Konfigurationsdatei.

Melden Sie den Status des CloudWatch Logs-Agenten

Verwenden Sie das folgende Verfahren, um den Status des CloudWatch Logs-Agenten auf Ihrer EC2-Instance zu melden.

So melden Sie den Status des Agenten

1. Stellen Sie eine Verbindung zu Ihrer EC2- Instance her. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

Weitere Informationen zu Verbindungsproblemen finden Sie unter [Troubleshooting Connecting to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch

2. Geben Sie als Eingabeaufforderung den folgenden Befehl ein:

```
sudo service awslogs status
```

Wenn Sie Amazon Linux 2 ausführen, geben Sie den folgenden Befehl ein:

```
sudo service awslogsd status
```

3. Überprüfen Sie die Datei `/var/log/awslogs.log` auf Fehler, Warnungen oder Probleme mit dem CloudWatch Logs-Agenten.

Starten Sie den CloudWatch Logs-Agent

Wenn der CloudWatch Logs-Agent auf Ihrer EC2-Instance nach der Installation nicht automatisch gestartet wurde oder wenn Sie den Agenten gestoppt haben, können Sie den Agenten mit dem folgenden Verfahren starten.

So starten Sie den -Agenten:

1. Stellen Sie eine Verbindung zu Ihrer EC2- Instance her. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

Weitere Informationen zu Verbindungsproblemen finden Sie unter [Troubleshooting Connecting to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

2. Geben Sie als Eingabeaufforderung den folgenden Befehl ein:

```
sudo service awslogs start
```

Wenn Sie Amazon Linux 2 ausführen, geben Sie den folgenden Befehl ein:

```
sudo service awslogsd start
```

Stoppen Sie den CloudWatch Logs-Agenten

Gehen Sie wie folgt vor, um den CloudWatch Logs-Agent auf Ihrer EC2-Instance zu beenden.

So beenden Sie den Agenten

1. Stellen Sie eine Verbindung zu Ihrer EC2- Instance her. Weitere Informationen finden Sie unter [Connect to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

Weitere Informationen zu Verbindungsproblemen finden Sie unter [Troubleshooting Connecting to Your Instance](#) im Amazon EC2 EC2-Benutzerhandbuch.

2. Geben Sie als Eingabeaufforderung den folgenden Befehl ein:

```
sudo service awslogs stop
```

Wenn Sie Amazon Linux 2 ausführen, geben Sie den folgenden Befehl ein:

```
sudo service awslogsd stop
```

CloudWatch Referenz zum Logs-Agenten

Important

Dieser Abschnitt ist eine Referenz für Benutzer des veralteten CloudWatch Logs-Agenten. Wenn Sie Instance Metadata Service Version 2 (IMDSv2) verwenden, müssen Sie den neuen Unified CloudWatch Agent verwenden. Auch wenn Sie ihn nicht verwenden, empfehlen wir dringend, den neueren Unified CloudWatch Agent anstelle des veralteten CloudWatch Logs-Agenten zu verwenden. Informationen zum neueren Unified CloudWatch Agent finden Sie unter [Erfassung von Metriken und Protokollen von Amazon EC2 EC2-Instances und lokalen Servern mit dem CloudWatch Agent](#). Informationen zur Migration vom veralteten CloudWatch Logs-Agenten zum Unified Agent finden Sie unter [Erstellen Sie die Agent-Konfigurationsdatei mit dem CloudWatch Assistenten](#).

Der CloudWatch Logs-Agent bietet eine automatisierte Möglichkeit, Protokolldaten von Amazon EC2 EC2-Instances an CloudWatch Logs zu senden. Der Agent enthält die folgenden Komponenten:

- Ein Plug-in für das AWS CLI, das Protokolldaten in Logs überträgt. CloudWatch
- Ein Skript (Daemon), das den Prozess initiiert, um Daten in Logs zu übertragen. CloudWatch
- Ein Cron-Auftrag, der sicherstellt, dass der Daemon ständig ausgeführt wird.

Agent-Konfigurationsdatei

Die CloudWatch Logs-Agent-Konfigurationsdatei beschreibt Informationen, die der CloudWatch Logs-Agent benötigt. Im Abschnitt [general] der Agent-Konfigurationsdatei werden gängige Konfigurationen definiert, die für alle Protokoll-Streams gelten. Im Abschnitt [logstream] werden die erforderlichen Informationen zum Senden einer lokalen Datei an einen Remote-Protokoll-Stream definiert. Sie können mehrere [logstream]-Abschnitte definieren, von denen aber jeder einen eindeutigen Namen innerhalb der Konfigurationsdatei haben muss, z. B. [logstream1], [logstream2] und so weiter. Der Wert von [logstream] sowie die erste Datenzeile in der Protokolldatei definieren die Identität der Protokolldatei.

```
[general]
state_file = value
logging_config_file = value
use_gzip_http_content_encoding = [true | false]
```

```
[logstream1]
log_group_name = value
log_stream_name = value
datetime_format = value
time_zone = [LOCAL|UTC]
file = value
file_fingerprint_lines = integer | integer-integer
multi_line_start_pattern = regex | {datetime_format}
initial_position = [start_of_file | end_of_file]
encoding = [ascii|utf_8|..]
buffer_duration = integer
batch_count = integer
batch_size = integer

[logstream2]
...
```

state_file

Gibt an, wo die Zustandsdatei gespeichert ist.

logging_config_file

(Optional) Gibt an, wo sich die Konfigurationsdatei für die Agent-Protokollierung befindet.

Wenn Sie keine Konfigurationsdatei für die Agent-Protokollierung angeben, wird immer die Standarddatei `awslogs.conf` verwendet. Der Standardspeicherort der Datei ist `/var/awslogs/etc/awslogs.conf`, wenn Sie den Agenten mit einem Skript installiert haben, und `/etc/awslogs/awslogs.conf`, wenn Sie den Agenten mit rpm installiert haben. Die Datei hat das Python-Konfigurationsdateiformat (<https://docs.pylogging-config-fileformatthon.org/2/library/logging.config.html#>). Logger mit den folgenden Namen können angepasst werden.

```

cwlogs.push
cwlogs.push.reader
cwlogs.push.publisher
cwlogs.push.event
cwlogs.push.batch
cwlogs.push.stream
cwlogs.push.watcher
```

Im unten stehenden Beispiel ändert sich die Ebene für Reader und Veröffentlicher auf **WARNUNG**, während der Standardwert **INFO** lautet.

```
[loggers]
keys=root,cwlogs,reader,publisher

[handlers]
keys=consoleHandler

[formatters]
keys=simpleFormatter

[logger_root]
level=INFO
handlers=consoleHandler

[logger_cwlogs]
level=INFO
handlers=consoleHandler
qualname=cwlogs.push
propagate=0

[logger_reader]
level=WARNING
handlers=consoleHandler
qualname=cwlogs.push.reader
propagate=0

[logger_publisher]
level=WARNING
handlers=consoleHandler
qualname=cwlogs.push.publisher
propagate=0

[handler_consoleHandler]
class=logging.StreamHandler
level=INFO
formatter=simpleFormatter
args=(sys.stderr,)

[formatter_simpleFormatter]
format=%(asctime)s - %(name)s - %(levelname)s - %(process)d - %(threadName)s -
%(message)s
```

use_gzip_http_content_encoding

Wenn sie auf true gesetzt ist (Standard), aktiviert sie die Gzip-HTTP-Inhaltskodierung, um komprimierte Nutzlasten an Logs zu senden. CloudWatch Dies verringert die CPU-Auslastung NetworkOut, senkt und verringert die Put-Latenz. Um diese Funktion zu deaktivieren, fügen Sie `use_gzip_http_content_encoding = false` zum Abschnitt [general] der CloudWatch Logs-Agent-Konfigurationsdatei hinzu und starten Sie den Agenten neu.

Note

Diese Einstellung ist nur awsccli-cwlogs Version 1.3.3 und höher verfügbar.

log_group_name

Gibt die Ziel-Protokollgruppe an. Eine Protokollgruppe wird automatisch erstellt, sofern diese nicht bereits vorhanden ist. Protokollgruppennamen können zwischen 1 und 512 Zeichen lang sein. Zulässige Zeichen sind a-z, A-Z, 0-9, '_' (Unterstrich), '-' (Bindestrich), '/' (Schrägstrich) und '.' (Zeitraum).

log_stream_name

Gibt den Ziel-Protokoll-Stream an. Sie können den Protokoll-Stream-Namen mithilfe einer Literalzeichenfolge oder den vordefinierten Variablen ({instance_id}, {hostname} und {ip_address}) oder einer Kombination aus beiden definieren. Ein Protokoll-Stream wird automatisch erstellt, sofern dieser nicht bereits vorhanden ist.

datetime_format

Gibt an, wie der Zeitstempel aus Protokollen extrahiert wird. Der Zeitstempel wird zum Abrufen von Protokollereignissen und Generieren von Metriken verwendet. Wenn `datetime_format` nicht angegeben ist, wird für die einzelnen Protokollereignisse die aktuelle Uhrzeit verwendet. Wenn der vorhandene Wert `datetime_format` für eine bestimmte Protokollnachricht ungültig ist, wird der Zeitstempel ab dem letzten Protokollereignis mit erfolgreich analysiertem Zeitstempel verwendet. Sind keine vorherigen Protokollereignisse vorhanden, wird die aktuelle Uhrzeit verwendet.

Die häufig verwendeten `datetime_format`-Codes sind unten aufgeführt. Sie können auch alle `datetime_format`-Codes verwenden, die von Python, `datetime.strptime()` unterstützt werden. Der Zeitzoneversatz (%z) wird ebenfalls unterstützt, wenn auch nur bis python 3.2, [+ -]HHMM ohne Doppelpunkt (:). Weitere Informationen finden Sie unter [strftime\(\)- und strptime \(\)-Verhalten](#).

%y: Jahr ohne Jahrhundert als mit Nullen aufgefüllte Dezimalzahl. 00, 01, ..., 99

%Y: Jahr mit Jahrhundert als Dezimalzahl. 1970, 1988, 2001, 2013

%b: Monat als regionale Abkürzung des Namens. Jan, Feb, ..., Dec (en_US);

%B: Vollständiger regionaler Name des Monats. January, February, ..., December (en_US);

%m: Monat als mit Nullen aufgefüllte Dezimalzahl. 01, 02, ..., 12

%d: Tag des Monats als mit Nullen aufgefüllte Dezimalzahl. 01, 02, ..., 31

%H: Stunde (24-Stunden) als mit Nullen aufgefüllte Dezimalzahl. 00, 01, ..., 23

%I: Stunde (12-Stunden) als mit Nullen aufgefüllte Dezimalzahl. 01, 02, ..., 12

%p: Regionales Äquivalent für AM oder PM.

%M: Minute als mit Nullen aufgefüllte Dezimalzahl. 00, 01, ..., 59

%S: Sekunde als mit Nullen aufgefüllte Dezimalzahl. 00, 01, ..., 59

%f: Mikrosekunde als links mit Nullen aufgefüllte Dezimalzahl. 000000, ..., 999999

%z: UTC-Verschiebung in Form von +HHMM oder -HHMM. +0000, -0400, +1030

Beispielformate:

Syslog: '%b %d %H:%M:%S', e.g. Jan 23 20:59:29

Log4j: '%d %b %Y %H:%M:%S', e.g. 24 Jan 2014 05:00:00

ISO8601: '%Y-%m-%dT%H:%M:%S%z', e.g. 2014-02-20T05:20:20+0000

time_zone

Gibt die Zeitzone des Zeitstempels des Protokollereignisses an. Die beiden unterstützten Werte sind UTC und LOCAL. Standardmäßig wird LOCAL verwendet, wenn die Zeitzone nicht von `datetime_format` abgeleitet werden kann.

file

Gibt Protokolldateien an, die Sie per Push in CloudWatch Logs speichern möchten. Die Datei kann auf eine bestimmte Datei oder mehrere Dateien verweisen (mithilfe von Platzhaltern wie/

var/log/system.log*). Nur die neueste Datei wird basierend auf der Änderungszeit der Datei in die CloudWatch Logs übertragen. Wir empfehlen, dass Sie eine Reihe von Platzhaltern angeben, z. B. Dateien desselben Typs, access_log.2014-06-01-01, access_log.2014-06-01-02 und so weiter, aber nicht mehrere Arten von Dateien, wie z. B. access_log_80 und access_log_443. Wenn Sie mehrere Arten von Dateien angeben möchten, fügen Sie der Konfigurationsdatei einen anderen Protokoll-Stream-Eintrag hinzu, damit jede Art von Protokolldatei in verschiedene Protokoll-Streams gestellt wird. Komprimierte Dateien werden nicht unterstützt.

file_fingerprint_lines

Gibt den Bereich von Zeilen an, über die eine Datei identifiziert wird. Gültige Werte sind eine Zahl oder zwei durch Gedankenstriche getrennten Zahlen, wie "1", "2-5". Der Standardwert ist 1, damit die erste Zeile für die Berechnung des Fingerabdrucks verwendet wird. Fingerabdruckzeilen werden nur dann an CloudWatch Logs gesendet, wenn alle angegebenen Zeilen verfügbar sind.

multi_line_start_pattern

Gibt das Muster an, anhand dessen der Beginn einer Protokolldatei identifiziert wird. Eine Protokollmeldung besteht aus einer Zeile, die mit dem angegebenen Muster übereinstimmt, und allen folgenden Zeilen, die nicht dem Muster entsprechen. Gültige Werte sind reguläre Ausdrücke oder {datetime_format}. Bei der Verwendung von {datetime_format} muss die Option "datetime_format" angegeben werden. Der Standardwert ist "^ [^\s]", sodass bei jeder Zeile, die mit Zeichen ohne Leerzeichen beginnt, die vorherige Protokollmeldung abgeschlossen wird und eine neue Protokollnachricht startet.

initial_position

Gibt an, wo der Anfang zum Lesen der Daten ist (start_of_file oder end_of_file). Der Standardwert ist start_of_file. Es wird nur verwendet, wenn für diesen Protokoll-Stream kein Zustand besteht.

encoding

Gibt die Verschlüsselung der Protokolldatei an, damit die Datei korrekt gelesen werden kann. Der Standardwert ist utf_8. Hier können von Python codecs.decode() unterstützte Verschlüsselungen verwendet werden.

Warning

Die Angabe einer fehlerhaften Kodierung kann zu Datenverlust führen, weil Zeichen, die nicht dekodiert werden können, durch ein anderes Zeichen ersetzt werden.

Im Folgenden sind einige gängige Kodierungen aufgeführt:

ascii, big5, big5hkscs, cp037, cp424, cp437, cp500, cp720, cp737, cp775, cp850, cp852, cp855, cp856, cp857, cp858, cp860, cp861, cp862, cp863, cp864, cp865, cp866, cp869, cp874, cp875, cp932, cp949, cp950, cp1006, cp1026, cp1140, cp1250, cp1251, cp1252, cp1253, cp1254, cp1255, cp1256, cp1257, cp1258, euc_jp, euc_jis_2004, euc_jisx0213, euc_kr, gb2312, gbk, gb18030, hz, iso2022_jp, iso2022_jp_1, iso2022_jp_2, iso2022_jp_2004, iso2022_jp_3, iso2022_jp_ext, iso2022_kr, latin_1, iso8859_2, iso8859_3, iso8859_4, iso8859_5, iso8859_6, iso8859_7, iso8859_8, iso8859_9, iso8859_10, iso8859_13, iso8859_14, iso8859_15, iso8859_16, johab, koi8_r, koi8_u, mac_cyrillic, mac_greek, mac_iceland, mac_latin2, mac_roman, mac_turkish, ptcp154, shift_jis, shift_jis_2004, shift_jisx0213, utf_32, utf_32_be, utf_32_le, utf_16, utf_16_be, utf_16_le, utf_7, utf_8, utf_8_sig

buffer_duration

Gibt die Dauer für die Stapelverarbeitung von Protokollereignissen an. Der Mindestwert und der Standardwert ist 5 000ms.

batch_count

Gibt die maximale Anzahl der Protokollereignisse in einem Stapel an, bis zu 10.000. Der Standardwert lautet 10.000.

batch_size

Gibt die maximale Größe von Protokollereignissen in einem Stapel in Byte an, bis zu 1048576 Byte. Der Standardwert lautet 1048576 Bytes. Diese Größe ist die Summe aller Ereignismeldungen in UTF-8 plus 26 Bytes pro Protokollereignis.

Verwenden des CloudWatch Logs-Agenten mit HTTP-Proxys

Sie können den CloudWatch Logs-Agent mit HTTP-Proxys verwenden.

Note

HTTP-Proxys werden awslogs-agent-setup in.py-Version 1.3.8 oder höher unterstützt.

Um den CloudWatch Logs-Agent mit HTTP-Proxys zu verwenden

1. Führen Sie eine der folgenden Aktionen aus:
 - a. Führen Sie für eine Neuinstallation des CloudWatch Logs-Agenten die folgenden Befehle aus:

```
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py -O
```

```
sudo python awslogs-agent-setup.py --region us-east-1 --http-proxy http://your/proxy --https-proxy http://your/proxy --no-proxy 169.254.169.254
```

Um den Zugriff auf den Amazon-EC2-Metadaten-Service für EC2-Instances zu behalten, verwenden Sie `--no-proxy 169.254.169.254` (empfohlen). Weitere Informationen finden Sie unter [Instance-Metadaten und Benutzerdaten](#) im Amazon EC2 EC2-Benutzerhandbuch.

Geben Sie in den Werten für `http-proxy` und `https-proxy` die gesamte URL ein.

- b. Für eine bestehende Installation des CloudWatch Logs-Agenten bearbeiten Sie `/var/awslogs/etc/proxy.conf` und fügen Sie Ihre Proxys hinzu:

```
HTTP_PROXY=  
HTTPS_PROXY=  
NO_PROXY=
```

2. Starten Sie den Agenten, damit die Änderungen wirksam werden:

```
sudo service awslogs restart
```

Wenn Sie Amazon Linux 2 verwenden, verwenden Sie den folgenden Befehl, um den Agenten neu zu starten:

```
sudo service awslogsd restart
```

Kompartimentierung der Logs-Agent-Konfigurationsdateien CloudWatch

Wenn `awslogs-agent-setup` Sie.py Version 1.3.8 oder höher mit `awscli-cwlogs` 1.3.3 oder höher verwenden, können Sie verschiedene Stream-Konfigurationen für verschiedene Komponenten

unabhängig voneinander importieren, indem Sie zusätzliche Konfigurationsdateien im Verzeichnis//erstellen. `var/awslogs/etc/config` Wenn der CloudWatch Logs-Agent gestartet wird, schließt er alle Stream-Konfigurationen in diese zusätzlichen Konfigurationsdateien ein. Die Konfigurationseigenschaften im Abschnitt `[general]` müssen in der Hauptkonfigurationsdatei (`/var/awslogs/etc/awslogs.conf`) and are ignored in any additional configuration files found in `/var/awslogs/etc/config/` definiert werden.

Wenn Sie kein Verzeichnis `var/awslogs/etc/config/` haben, weil Sie den Agenten mit `rpm` installiert haben, können Sie stattdessen das Verzeichnis `etc/awslogs/config/` verwenden.

Starten Sie den Agenten, damit die Änderungen wirksam werden:

```
sudo service awslogs restart
```

Wenn Sie Amazon Linux 2 verwenden, verwenden Sie den folgenden Befehl, um den Agenten neu zu starten:

```
sudo service awslogsd restart
```

CloudWatch Häufig gestellte Fragen zum Log-Agenten

Welche Arten von Dateirotationen werden unterstützt?

Folgende Dateirotationsmechanismen werden unterstützt:

- Umbenennen vorhandener Protokolldateien mit einem numerischen Suffix, anschließendes Neuerstellen der ursprünglichen leeren Protokolldatei. Zum Beispiel `/var/log/syslog.log` is renamed `/var/log/syslog.log.1`. If `/var/log/syslog.log.1` already exists from a previous rotation, it is renamed `/var/log/syslog.log.2`.
- Kürzen der ursprünglichen Protokolldatei nach der Erstellung einer Kopie. Zum Beispiel ist `/var/log/syslog.log` is copied to `/var/log/syslog.log.1` and `/var/log/syslog.log` gekürzt. In diesem Fall können Datenverluste entstehen. Verwenden Sie diesen Dateirotationsmechanismus also mit Bedacht.
- Erstellen einer neuen Datei mit dem gemeinsamen Muster der alten Datenbank. Zum Beispiel wird `/var/log/syslog.log.2014-01-01` remains and `/var/log/syslog.log.2014-01-02` erstellt.

Der Fingerabdruck (Quell-ID) der Datei wird berechnet, indem ein Hash für den Protokoll-Stream-Schlüssel und die erste Zeile in der Datei durchgeführt wird. Zum Überschreiben dieses

Verhaltens kann die Option `file_fingerprint_lines` verwendet werden. Bei einer Rotation sollte die neue Datei neue Inhalte haben, und die alte Datei sollte keine angehängten Inhalte haben. Der Agent überträgt die neue Datei, wenn der Lesevorgang der alten Datei abgeschlossen ist.

Wie kann ich bestimmen, welche Agent-Version ich verwende?

Wenn Sie zur Installation des CloudWatch Logs-Agenten ein Setup-Skript verwendet haben, können Sie mit `mit/var/awslogs/bin/awslogs-version.sh` überprüfen, welche Version des Agenten Sie verwenden. Dabei werden die Version des Agent und seine großen Abhängigkeiten gedruckt. Wenn Sie Yum zur Installation des CloudWatch Logs-Agenten verwendet haben, können Sie „yum info awslogs“ und „yum info aws-cli-plugin-cloudwatch -logs“ verwenden, um die Version des Logs-Agenten und des Plugins zu überprüfen. CloudWatch

Wie werden Protokolleinträge in Protokollereignisse umgewandelt?

Protokollereignisse enthalten zwei Eigenschaften: den Zeitstempel mit Datum und Uhrzeit des Ereignisses und die reine Protokollnachricht. Standardmäßig wird bei jeder Zeile, die mit Zeichen ohne Leerzeichen beginnt, die vorherige Protokollnachricht (falls vorhanden) abgeschlossen und eine neue Protokollnachricht gestartet. Um dieses Verhalten zu überschreiben, kann `multi_line_start_pattern` verwendet werden, und bei jeder Zeile, die mit dem Muster übereinstimmt, wird eine neue Protokollnachricht gestartet. Das Muster ist ein regulärer Ausdruck oder `'{datetime_format}'`. Wenn beispielsweise die erste Zeile in jeder Protokollnachricht einen Zeitstempel wie z. B. "2014-01-02T13:13:01Z" enthält, kann das `multi_line_start_pattern` auf `"\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}Z"` gesetzt werden. Zum Vereinfachen der Konfiguration, kann die Variable `'{datetime_format}'` verwendet werden, wenn `datetime_format` option angegeben ist. Für dasselbe Beispiel gilt, wenn `datetime_format` auf `'%Y-%m-%dT%H:%M:%S%z'` gesetzt ist, dann könnte `multi_line_start_pattern` einfach `'{datetime_format}'` sein.

Wenn `datetime_format` nicht angegeben ist, wird für die einzelnen Protokollereignisse die aktuelle Uhrzeit verwendet. Wenn der vorhandene Wert `datetime_format` für eine bestimmte Protokollnachricht ungültig ist, wird der Zeitstempel ab dem letzten Protokollereignis mit erfolgreich analysiertem Zeitstempel verwendet. Sind keine vorherigen Protokollereignisse vorhanden, wird die aktuelle Uhrzeit verwendet. Eine Warnmeldung wird protokolliert, wenn ein Protokollereignis auf die aktuelle Uhrzeit oder den Zeitpunkt des vorherigen Protokollereignisses zurückgreift.

Zeitstempel dienen zum Abrufen von Protokollereignissen und Generieren von Metriken. Wenn Sie das falsche Format angeben, könnten die Protokollereignisse nicht abrufbar werden, und es werden falsche Metriken generiert.

Wie werden Protokollereignisse im Stapel verarbeitet?

Eine Stapel ist voll und wird veröffentlicht, wenn eine der folgenden Bedingungen erfüllt ist:

1. Der Zeitraum `buffer_duration` ist abgelaufen, nachdem das erste Protokollereignis hinzugefügt wurde.
2. Weniger als `batch_size` der Protokollereignisse wurden akkumuliert, aber durch das Hinzufügen des neuen Protokollereignisses wird `batch_size` überschritten.
3. Die Anzahl der Protokollereignisse hat `batch_count` erreicht.
4. Protokollereignisse aus dem Stapel umfassen nie mehr als 24 Stunden. Aber durch das Hinzufügen des neuen Protokollereignisses wird die 24-Stunden-Bedingung überschritten.

Wodurch werden Protokolleinträge, Protokollereignisse oder Stapel übersprungen oder gekürzt?

Zur Einhaltung der Bedingung für die `PutLogEvents`-Operation können folgende Probleme dazu führen, dass ein Protokollereignis oder ein Stapel übersprungen wird.

Note

Der CloudWatch Logs-Agent schreibt eine Warnung in sein Protokoll, wenn Daten übersprungen werden.

1. Wenn das Protokollereignis größer als 256 KB ist, wird es vollständig übersprungen.
2. Wenn der Zeitstempel des Protokollereignisses mehr als 2 Stunden in der Zukunft liegt, wird das Protokollereignis übersprungen.
3. Wenn der Zeitstempel des Protokollereignisses mehr als 14 Stunden in der Vergangenheit liegt, wird das Protokollereignis übersprungen.
4. Wenn ein Protokollereignis älter als der Aufbewahrungszeitraum der Protokollgruppe ist, wird der gesamte Stapel übersprungen.
5. Wenn der Stapel von Protokollereignissen in einer einzigen `PutLogEvents`-Anforderung mehr als 24 Stunden umfasst, schlägt die `PutLogEvents`-Operation fehl.

Führt das Anhalten des Agent zu Datenverlust/Duplikaten?

Nicht, solange die Zustandsdatei verfügbar ist und seit der letzten Ausführung keine Dateirotation stattgefunden hat. Der CloudWatch Logs-Agent kann an der Stelle beginnen, an der er aufgehört hat, und die Protokolldaten weiterleiten.

Kann ich verschiedene Protokolldateien aus demselben Host oder aus unterschiedlichen Hosts demselben Protokoll-Stream zuweisen?

Die Konfiguration von mehreren Protokollquellen, damit Daten an einen einzelnen Protokoll-Stream gesendet werden, wird nicht unterstützt.

Welche API-Aufrufe führt der Agent durch (oder welche Aktionen sollte ich meiner IAM-Richtlinie hinzufügen)?

Der CloudWatch Logs-Agent benötigt die Erlaubnis `CreateLogGroup`, `CreateLogStream`, `DescribeLogStreams`, `DescribeLogGroups`, `PutLogEvents` und `PutRetentionPolicy` Aktionen auszuführen. Wenn Sie den neuesten Agent verwendet, wird `DescribeLogStreams` nicht mehr benötigt. Weitere Informationen finden Sie im Beispiel für eine IAM-Richtlinie unten.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "logs:DescribeLogGroups",
        "logs:PutRetentionPolicy"
      ],
      "Resource": [
        "arn:aws:logs:*:*:*"
      ]
    }
  ]
}
```

Ich möchte nicht, dass der CloudWatch Logs-Agent automatisch Protokollgruppen oder Protokollstreams erstellt. Wie kann ich verhindern, dass der Agent Protokollgruppen und Protokoll-Streams neu erstellt?

Sie können den Agent in Ihrer IAM-Richtlinie auf die folgenden Operationen beschränken: `DescribeLogStreams`, `PutLogEvents`.

Bevor Sie die Berechtigungen `CreateLogGroup` und `CreateLogStream` vom Agenten entziehen, müssen Sie sowohl die Protokollgruppen als auch die Protokolldatenströme erstellen, die der Agent verwenden soll. Der Protokoll-Agent kann keine Protokolldatenströme in einer Protokollgruppe erstellen, die Sie erstellt haben, es sei denn, er verfügt über die Berechtigungen `CreateLogGroup` und `CreateLogStream`.

Welche Protokolle sollte ich bei der Fehlerbehebung beachten?

Das Installationsprotokoll des Agenten finden Sie unter `/var/log/awslogs-agent-setup.log` und das Protokoll des Agenten unter `/var/log/awslogs.log`.

Schnellstart: Verwenden Sie diese Option CloudFormation , um mit CloudWatch Logs zu beginnen

AWS CloudFormation ermöglicht es Ihnen, Ihre AWS Ressourcen im JSON-Format zu beschreiben und bereitzustellen. Zu den Vorteilen dieser Methode gehören die Möglichkeit, eine Sammlung von AWS Ressourcen als eine einzige Einheit zu verwalten und Ihre AWS Ressourcen problemlos regionsübergreifend zu replizieren.

Bei der Bereitstellung AWS mithilfe von erstellen Sie Vorlagen CloudFormation, in denen die zu verwendenden AWS Ressourcen beschrieben werden. Das folgende Beispiel ist ein Vorlagenausschnitt, mit dem eine Protokollgruppe und ein Metrikfilter erstellt werden, der 404 Vorkommnisse zählt und diese Zählung an die Protokollgruppe sendet.

```
"WebServerLogGroup": {
  "Type": "AWS::Logs::LogGroup",
  "Properties": {
    "RetentionInDays": 7
  }
},

"404MetricFilter": {
  "Type": "AWS::Logs::MetricFilter",
  "Properties": {
    "LogGroupName": {
      "Ref": "WebServerLogGroup"
    },
    "FilterPattern": "[ip, identity, user_id, timestamp, request, status_code = 404, size, ...]",
    "MetricTransformations": [
```

```
{
  "MetricValue": "1",
  "MetricNamespace": "test/404s",
  "MetricName": "test404Count"
}
]
```

Dies ist ein einfaches Beispiel. Mithilfe von können Sie viel umfangreichere CloudWatch Logs-Bereitstellungen einrichten. CloudFormation Weitere Informationen zu Vorlagenbeispielen finden Sie unter [Amazon CloudWatch Logs Template Snippets](#) im AWS CloudFormation Benutzerhandbuch. Weitere Informationen zu den ersten Schritten finden Sie unter [Erste Schritte mit AWS CloudFormation](#) im AWS CloudFormation -Benutzerhandbuch.

Erfassung von Protokollen über HTTP-Endpunkte

Amazon CloudWatch Logs bietet HTTP-Endpunkte, mit denen Sie CloudWatch Protokolle mithilfe einfacher HTTP-POST-Anfragen direkt an Logs senden können. Diese Endpunkte unterstützen sowohl die SigV4-Authentifizierung als auch die Bearer-Token-Authentifizierung.

Important

Wir empfehlen die Verwendung der SigV4-Authentifizierung für alle Produktionsworkloads, bei denen AWS eine SDK-Integration möglich ist. SigV4 verwendet kurzfristige Anmeldeinformationen und bietet die stärkste Sicherheitslage. Die Bearer-Token-Authentifizierung (API-Schlüssel) ist für Szenarien vorgesehen, in denen SigV4 nicht durchführbar ist, z. B. für Log-Forwarder von Drittanbietern, die die SDK-Integration nicht unterstützen. AWS Weitere Informationen finden Sie unter [Alternativen zu langfristigen Zugriffsschlüsseln](#) im IAM-Benutzerhandbuch.

CloudWatch Logs unterstützt die folgenden Endpunkte für die HTTP-Erfassung:

Endpoint	Pfad	Content-Type	Format
OpenTelemetry Logs	/v1/logs	application/json oder application/x-protobuf	OTLP-JSON oder Protobuf
HLC Logs	/services/collector/event	application/json	HLC-Format
ND-JSON Logs	/ingest/bulk	application/json oder application/x-ndjson	JSON
Structured JSON Logs	/ingest/json	application/json	JSON-Objekt oder -Array

Gemeinsames Verhalten

Alle HTTP-Ingestion-Endpunkte weisen das folgende Verhalten auf:

Authentifizierung

Alle Endpunkte unterstützen sowohl die SigV4-Authentifizierung als auch die Bearer-Token-Authentifizierung:

- Sigv4 (empfohlen) — Signierung mit AWS Standardsignatur Version 4. Verwenden Sie SigV4 immer dann, wenn Ihre Anwendung oder Infrastruktur das AWS SDK unterstützt oder Anfragen signieren kann. SigV4 verwendet kurzfristige Anmeldeinformationen und ist die sicherste Authentifizierungsmethode.
- Bearer-Token — Verwenden Sie den `Authorization: Bearer <ACWL token>` Header.
 - Das Token muss ein gültiges ACWL-Bearer-Token sein. Anweisungen zur Einrichtung finden Sie unter [the section called “Bearer-Token-Authentifizierung”](#)
 - Erfordert die `logs:PutLogEvents` und `logs:CallWithBearerToken` IAM-Berechtigungen.

Protokollgruppe und Protokollstream

- Wird über Header bereitgestellt: `x-aws-log-group` und `x-aws-log-stream`
- Abfrageparameter `?logGroup=<name>&logStream=<name>` werden auch auf allen Endpunkten außer OTLP unterstützt.
- Sie können nicht sowohl Abfrageparameter als auch Header für denselben Parameter verwenden.
- Sowohl die Protokollgruppe als auch der Protokollstream sind erforderlich.

Antwort

- Erfolg: HTTP 200 mit Körper {}
- Fehler bei der Validierung: HTTP 400
- Fehler bei der Authentifizierung: HTTP 401

Einrichtung der Bearer-Token-Authentifizierung

Bevor Sie Protokolle mithilfe der Bearer-Token-Authentifizierung mit einem der HTTP-Ingestion-Endpunkte senden können, müssen Sie:

- Erstellen Sie einen IAM-Benutzer mit Logs-Berechtigungen CloudWatch
- Generieren Sie dienstspezifische Anmeldeinformationen (Bearer-Token)
- Erstellen Sie eine Protokollgruppe und einen Protokollstream
- Aktivieren Sie die Bearer-Token-Authentifizierung für die Protokollgruppe

Important

Wir empfehlen, für alle Workloads, bei denen dies möglich ist, die SigV4-Authentifizierung mit kurzfristigen Anmeldeinformationen zu verwenden. SigV4 bietet die stärkste Sicherheitslage. Beschränken Sie die Verwendung von API-Schlüsseln (Bearer-Token) auf Szenarien, in denen eine kurzfristige Authentifizierung auf der Grundlage von Anmeldeinformationen nicht möglich ist. Wenn Sie bereit sind, CloudWatch Logs in Anwendungen mit höheren Sicherheitsanforderungen zu integrieren, sollten Sie auf kurzfristige Anmeldedaten umsteigen. Weitere Informationen finden Sie unter [Alternativen zu langfristigen Zugriffsschlüsseln](#) im IAM-Benutzerhandbuch.

Option 1: Schnellstart mit der Konsole AWS

Die AWS Management Console bietet einen optimierten Arbeitsablauf zur Generierung von API-Schlüsseln für den HTTP-Endpunktzugriff.

So richten Sie den HTTP-Endpunktzugriff mithilfe der Konsole ein

1. Melden Sie sich bei der AWS Management Console an.
2. Navigieren Sie zu CloudWatch> Einstellungen > Protokolle.
3. Wählen Sie im Abschnitt API-Schlüssel die Option API-Schlüssel generieren aus.
4. Für den Ablauf des API-Schlüssels haben Sie folgende Möglichkeiten:
 - Wählen Sie eine Ablaufdauer für den API-Schlüssel von 1, 5, 30, 90 oder 365 Tagen aus.
 - Wählen Sie Benutzerdefinierte Dauer aus, um ein benutzerdefiniertes Ablaufdatum für den API-Schlüssel festzulegen.
 - Wählen Sie Nie abläuft aus (nicht empfohlen).
5. Wählen Sie API-Schlüssel generieren aus.

Die Konsole automatisch:

- Erstellt einen neuen IAM-Benutzer mit den entsprechenden Berechtigungen
 - Hängt die verwaltete [CloudWatchLogsAPIKeyAccess-Richtlinie](#) an (einschließlich `logs:PutLogEvents` und `logs:CallWithBearerToken` Berechtigungen)
 - Generiert dienstspezifische Anmeldeinformationen (API-Schlüssel)
6. Kopieren Sie die angezeigten Anmeldeinformationen und speichern Sie sie sicher:
- API-Schlüssel-ID (dienstspezifische Anmeldeinformations-ID)
 - Geheimer API-Schlüssel (Inhaber-Token)

 Important

Speichern Sie das API-Schlüsselgeheimnis sofort. Er kann später nicht mehr abgerufen werden. Wenn Sie ihn verlieren, müssen Sie einen neuen API-Schlüssel generieren.

7. Erstellen Sie die Protokollgruppe und den Protokollstream, in denen Ihre Protokolle gespeichert werden:

```
# Create the log group
aws logs create-log-group \
  --log-group-name /aws/hlc-logs/my-application \
  --region us-east-1

# Create the log stream
aws logs create-log-stream \
  --log-group-name /aws/hlc-logs/my-application \
  --log-stream-name application-stream-001 \
  --region us-east-1
```

8. Aktivieren Sie die Bearer-Token-Authentifizierung für die Protokollgruppe:

```
aws logs put-bearer-token-authentication \
  --log-group-identifier /aws/hlc-logs/my-application \
  --bearer-token-authentication-enabled \
  --region us-east-1
```

Überprüfen Sie die Konfiguration:

```
aws logs describe-log-groups \
```

```
--log-group-name-prefix /aws/hlc-logs/my-application \  
--region us-east-1
```

Enthaltene Berechtigungen: Der automatisch erstellte IAM-Benutzer verfügt über die folgenden Berechtigungen:

- `logs:PutLogEvents`— Sendet Protokollereignisse an CloudWatch Logs
- `logs:CallWithBearerToken`— Authentifizieren Sie sich mit dem Bearer-Token
- `kms:Describe*`, `kms:GenerateDataKey*`, `kms:Decrypt` — Greifen Sie auf KMS-verschlüsselte Protokollgruppen zu (mit einer Bedingung, die auf den Protokolldienst beschränkt ist)

Option 2: Manuelle Einrichtung

Wenn Sie mehr Kontrolle über die IAM-Konfiguration bevorzugen oder die Berechtigungen anpassen müssen, können Sie den HTTP-Endpunktzugriff manuell einrichten.

Schritt 1: Erstellen Sie einen IAM-Benutzer

Erstellen Sie einen IAM-Benutzer, der für die Protokollaufnahme verwendet wird:

1. Melden Sie sich bei der AWS Management Console an und navigieren Sie zu IAM.
2. Wählen Sie im linken Navigationsbereich Benutzer aus.
3. Wählen Sie Create user (Benutzer erstellen) aus.
4. Geben Sie einen Benutzernamen ein (z. B. `cloudwatch-logs-hlc-user`).
5. Wählen Sie Weiter aus.
6. Fügen Sie eine der folgenden IAM-Richtlinien hinzu:

Option A: Verwenden Sie die verwaltete Richtlinie (empfohlen)

Hängen Sie die verwaltete [CloudWatchLogsAPIKeyAccess-Richtlinie](#) an.

Option B: Erstellen Sie eine benutzerdefinierte Richtlinie

Erstellen Sie die folgende IAM-Richtlinie und fügen Sie sie an:

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "LogsAPIs",
    "Effect": "Allow",
    "Action": [
      "logs:CallWithBearerToken",
      "logs:PutLogEvents"
    ],
    "Resource": "*"
  },
  {
    "Sid": "KMSAPIs",
    "Effect": "Allow",
    "Action": [
      "kms:Describe*",
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Condition": {
      "StringEquals": {
        "kms:ViaService": [
          "logs.*.amazonaws.com"
        ]
      }
    },
    "Resource": "arn:aws:kms:*:*:key/*"
  }
]
```

7. Wählen Sie Weiter und dann Benutzer erstellen.

 Note

Die KMS-Berechtigungen sind erforderlich, wenn Sie Protokolle an KMS-verschlüsselte Protokollgruppen senden möchten. Die Bedingung beschränkt den KMS-Zugriff nur auf Schlüssel, die über CloudWatch den Logs-Dienst verwendet werden.

Schritt 2: Generieren Sie dienstspezifische Anmeldeinformationen (API-Schlüssel)

Generieren Sie den CloudWatch Logs-API-Schlüssel mithilfe der [CreateServiceSpecificCredential](#) API. Sie können auch den [create-service-specific-credential](#) CLI-Befehl verwenden. Für das Alter der Anmeldeinformationen können Sie einen Wert zwischen 1–36600 Tagen angeben. Wenn Sie keine Gültigkeitsdauer für die Anmeldeinformationen angeben, läuft der API-Schlüssel nicht ab.

Um einen API-Schlüssel mit einem Ablauf von 30 Tagen zu generieren:

```
aws iam create-service-specific-credential \  
  --user-name cloudwatch-logs-hlc-user \  
  --service-name logs.amazonaws.com \  
  --credential-age-days 30
```

Die Antwort ist ein [ServiceSpecificCredential](#) Objekt. Der `ServiceCredentialSecret` Wert ist Ihr CloudWatch Logs-API-Schlüssel (Bearer-Token).

Important

Speichern Sie den Wert „`ServiceCredentialSecret`“ sicher, da Sie ihn später nicht mehr abrufen können. Wenn Sie ihn verlieren, müssen Sie einen neuen API-Schlüssel generieren.

Schritt 3: Erstellen Sie eine Protokollgruppe und einen Protokollstream

Erstellen Sie die Protokollgruppe und den Protokollstream, in denen Ihre Protokolle gespeichert werden:

```
# Create the log group  
aws logs create-log-group \  
  --log-group-name /aws/hlc-logs/my-application \  
  --region us-east-1  
  
# Create the log stream  
aws logs create-log-stream \  
  --log-group-name /aws/hlc-logs/my-application \  
  --log-stream-name application-stream-001 \  
  --region us-east-1
```

Schritt 4: Aktivieren Sie die Bearer-Token-Authentifizierung

Aktivieren Sie die Bearer-Token-Authentifizierung für die Protokollgruppe:

```
aws logs put-bearer-token-authentication \  
  --log-group-identifier /aws/hlc-logs/my-application \  
  --bearer-token-authentication-enabled \  
  --region us-east-1
```

Überprüfen Sie die Konfiguration:

```
aws logs describe-log-groups \  
  --log-group-name-prefix /aws/hlc-logs/my-application \  
  --region us-east-1
```

Steuern Sie die Berechtigungen für die Generierung und Verwendung von CloudWatch Logs-API-Schlüsseln

Die Generierung und Verwendung von CloudWatch Logs-API-Schlüsseln wird durch Aktionen und Bedingungsschlüssel sowohl in den CloudWatch Logs- als auch in den IAM-Diensten gesteuert.

Steuerung der Generierung von CloudWatch Logs-API-Schlüsseln

Die `CreateServiceSpecificCredential` Aktion [iam:](#) steuert die Generierung eines dienstspezifischen Schlüssels (z. B. eines CloudWatch Logs-API-Schlüssels). Sie können diese Aktion auf IAM-Benutzer als Ressource beschränken, um die Anzahl der Benutzer einzuschränken, für die ein Schlüssel generiert werden kann.

Mit den folgenden Bedingungsschlüsseln können Sie Bedingungen mit der Berechtigung für die Aktion `iam:CreateServiceSpecificCredential` verknüpfen:

- [iam: ServiceSpecificCredentialAgeDays](#) — Ermöglicht es Ihnen, in der Bedingung die Ablaufzeit des Schlüssels in Tagen anzugeben. Sie können diesen Bedingungsschlüssel zum Beispiel verwenden, um nur das Erstellen von API-Schlüsseln zu erlauben, die innerhalb von 90 Tagen ablaufen.
- [iam: ServiceSpecificCredentialServiceName](#) — Ermöglicht es Ihnen, in der Bedingung den Namen eines Dienstes anzugeben. Sie können diesen Bedingungsschlüssel beispielsweise verwenden, um nur die Erstellung von API-Schlüsseln für CloudWatch Logs und nicht für andere Dienste zuzulassen.

Steuerung der Verwendung von CloudWatch Logs-API-Schlüsseln

Die `logs:CallWithBearerToken` Aktion steuert die Verwendung eines CloudWatch Logs-API-Schlüssels. Um zu verhindern, dass eine Identität CloudWatch Logs-API-Schlüssel verwendet, fügen Sie eine Richtlinie hinzu, die dem IAM-Benutzer, der dem Schlüssel zugeordnet ist, die `logs:CallWithBearerToken` Aktion verweigert.

Beispielrichtlinien

Verhindern Sie, dass eine Identität CloudWatch Logs-API-Schlüssel generiert und verwendet

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyCWLAPIKeys",
      "Effect": "Deny",
      "Action": [
        "iam:CreateServiceSpecificCredential",
        "logs:CallWithBearerToken"
      ],
      "Resource": "*"
    }
  ]
}
```

Warning

Diese Richtlinie verhindert die Erstellung von Anmeldeinformationen für alle AWS Dienste, die die Erstellung dienstspezifischer Anmeldeinformationen unterstützen. Weitere Informationen finden Sie unter [Servicespezifische Anmeldeinformationen für IAM-Benutzer](#).

Verhindern Sie, dass eine Identität CloudWatch Logs-API-Schlüssel verwendet

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "logs:CallWithBearerToken",
```



```

        "Resource": "*"
    }
]
}

```

Erlauben Sie die Erstellung von CloudWatch Logs-Schlüsseln nur, wenn sie innerhalb von 90 Tagen ablaufen

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceSpecificCredential",
      "Resource": "arn:aws:iam::123456789012:user/username",
      "Condition": {
        "StringEquals": {
          "iam:ServiceSpecificCredentialServiceName": "logs.amazonaws.com"
        },
        "NumericLessThanEquals": {
          "iam:ServiceSpecificCredentialAgeDays": "90"
        }
      }
    }
  ]
}

```

Rotierende API-Schlüssel

Durch die regelmäßige Rotation Ihrer API-Schlüssel wird das Risiko eines unbefugten Zugriffs verringert. Wir empfehlen, einen Rotationsplan aufzustellen, der den Sicherheitsrichtlinien Ihres Unternehmens entspricht.

Rotationsprozess

Gehen Sie wie folgt vor, um einen API-Schlüssel zu rotieren, ohne die Protokollzustellung zu unterbrechen:

1. Erstellen Sie einen neuen (sekundären) Berechtigungsnachweis für den IAM-Benutzer:

```
aws iam create-service-specific-credential \
```

```
--user-name cloudwatch-logs-hlc-user \  
--service-name logs.amazonaws.com \  
--credential-age-days 90
```

2. (Optional) Speichern Sie die neuen Anmeldeinformationen AWS Secrets Manager für einen sicheren Abruf und eine automatische Rotation.
3. Importieren Sie die neuen Anmeldeinformationen in das Portal Ihres Anbieters oder aktualisieren Sie Ihre Anwendungskonfiguration, um den neuen API-Schlüssel zu verwenden.
4. Setzen Sie die ursprünglichen Anmeldeinformationen auf inaktiv:

```
aws iam update-service-specific-credential \  
  --user-name cloudwatch-logs-hlc-user \  
  --service-specific-credential-id ACCA1234EXAMPLE1234 \  
  --status Inactive
```

5. Stellen Sie sicher, dass die Protokollzustellung nicht beeinträchtigt wird, indem Sie die IncomingBytes Metrik für Ihre Protokollgruppe in überwachen. CloudWatch Weitere Informationen finden Sie unter [Überwachung mit CloudWatch Metriken](#).
6. Nachdem Sie die erfolgreiche Zustellung mit dem neuen Schlüssel bestätigt haben, löschen Sie die vorherigen Anmeldeinformationen:

```
aws iam delete-service-specific-credential \  
  --service-specific-credential-id ACCA1234EXAMPLE1234
```

Ablauf des Schlüssels überwachen

Verwenden Sie den folgenden [list-service-specific-credentials](#) Befehl, um das Erstellungsdatum und den Status Ihrer vorhandenen API-Schlüssel zu überprüfen:

```
aws iam list-service-specific-credentials \  
  --user-name cloudwatch-logs-hlc-user \  
  --service-name logs.amazonaws.com
```

Die Antwort beinhaltet CreateDate und Status für jeden Berechtigungsnachweis. Verwenden Sie diese Informationen, um Schlüssel zu identifizieren, die bald ablaufen oder die schon länger aktiv sind, als es Ihre Rotationsrichtlinie zulässt.

Reaktion auf einen kompromittierten API-Schlüssel

Wenn Sie vermuten, dass ein API-Schlüssel kompromittiert wurde, führen Sie sofort die folgenden Schritte durch:

1. Deaktivieren Sie den Schlüssel sofort, um eine weitere unbefugte Verwendung zu verhindern:

```
aws iam update-service-specific-credential \  
  --user-name cloudwatch-logs-hlc-user \  
  --service-specific-credential-id ACCA1234EXAMPLE1234 \  
  --status Inactive
```

2. Überprüfen Sie die CloudTrail Protokolle, um den Umfang des unbefugten Zugriffs zu ermitteln. Weitere Informationen [the section called “Protokollierung der Verwendung von API-Schlüsseln mit CloudTrail”](#) zur Aktivierung der Überwachung der API-Schlüsselnutzung finden Sie unter.
3. Erstellen Sie nach dem unter beschriebenen Rotationsprozess einen Ersatzschlüssel [the section called “Rotationsprozess”](#).
4. Löschen Sie den kompromittierten Schlüssel, nachdem der Ersatzschlüssel installiert wurde:

```
aws iam delete-service-specific-credential \  
  --service-specific-credential-id ACCA1234EXAMPLE1234
```

5. Fügen Sie eine Ablehnungsrichtlinie hinzu, wenn Sie sofort den gesamten Zugriff auf Inhaber-Tokens für den IAM-Benutzer sperren müssen, während Sie die folgenden Untersuchungen durchführen:

```
{  
  "Version": "2012-10-17",  
  "Statement": {  
    "Effect": "Deny",  
    "Action": "logs:CallWithBearerToken",  
    "Resource": "*"   
  }  
}
```

 Note

Um diese Aktionen über die API auszuführen, müssen Sie sich mit AWS Anmeldeinformationen und nicht mit einem CloudWatch Logs-API-Schlüssel authentifizieren.

Sie können auch die folgenden IAM-API-Operationen verwenden, um kompromittierte Schlüssel zu verwalten:

- [ResetServiceSpecificCredential](#)— Setzen Sie den Schlüssel zurück, um ein neues Passwort zu generieren, ohne die Anmeldeinformationen zu löschen. Der Schlüssel darf nicht abgelaufen sein.

Bewährte Sicherheitsmethoden für API-Schlüssel

Folgen Sie diesen bewährten Methoden, um Ihre CloudWatch Logs-API-Schlüssel zu schützen:

- Betten Sie niemals API-Schlüssel in den Quellcode ein. Codieren Sie API-Schlüssel nicht fest im Anwendungscode und übergeben Sie sie nicht an Versionskontrollsysteme. Wenn ein Schlüssel versehentlich in ein öffentliches Repository übertragen wird, wird er möglicherweise AWS durch automatisiertes Scannen gekennzeichnet. Sie sollten den Schlüssel sofort wechseln.
- Verwenden Sie einen Secrets-Manager. Speichern Sie API-Schlüssel in [AWS Secrets Manager](#) oder einer gleichwertigen Lösung zur Verwaltung von Geheimnissen. Dies ermöglicht eine zentrale Zugriffskontrolle, Auditprotokollierung und automatisierte Rotation.
- Legen Sie für alle Schlüssel ein Ablaufdatum fest. Geben Sie beim Erstellen von API-Schlüsseln immer einen `--credential-age-days` Wert an. Verwenden Sie den `iam:ServiceSpecificCredentialAgeDays` IAM-Bedingungsschlüssel, um eine maximale Gültigkeitsdauer von Schlüsseln in Ihrer Organisation durchzusetzen. Ein Beispiel finden Sie unter [the section called “Erlauben Sie die Erstellung von CloudWatch Logs-Schlüsseln nur, wenn sie innerhalb von 90 Tagen ablaufen”](#).
- Wenden Sie Berechtigungen mit den geringsten Rechten an. Beschränken Sie die Berechtigungen des IAM-Benutzers nur auf die erforderlichen Protokollgruppen und Aktionen. Verwenden Sie die Managed [CloudWatchLogsAPIKeyAccess-Richtlinie](#) als Ausgangspunkt und schränken Sie sie bei Bedarf weiter ein.
- Aktivieren Sie die CloudTrail Protokollierung. Überwachen Sie die Verwendung von API-Schlüsseln, indem Sie CloudTrail Datenereignisse für

aktivieren `AWS::Logs::LogGroupAuthorization`. Siehe [the section called “Protokollierung der Verwendung von API-Schlüsseln mit CloudTrail”](#).

- Überwachen Sie mit IAM Access Analyzer. Verwenden Sie [IAM Access Analyzer](#), um ungenutzte Anmeldeinformationen und übermäßig freizügige Richtlinien im Zusammenhang mit Ihren API-Schlüssel-IAM-Benutzern zu identifizieren.
- Wechseln Sie die Schlüssel regelmäßig. Legen Sie einen Rotationsplan fest und folgen Sie dem unter beschriebenen Verfahren [the section called “Rotierende API-Schlüssel”](#).

Protokollierung der Verwendung von API-Schlüsseln mit CloudTrail

Sie können AWS CloudTrail es verwenden, um Datenereignisse für die Verwendung von CloudWatch Logs API-Schlüsseln zu protokollieren. CloudWatch Logs sendet `AWS::Logs::LogGroupAuthorization` Datenereignisse für `CallWithBearerToken` Aufrufe aus, sodass Sie überprüfen können, wann und wie API-Schlüssel zum Senden von Protokollen verwendet werden.

So aktivieren Sie die CloudTrail Protokollierung für die Verwendung von CloudWatch Logs-API-Schlüsseln:

Note

Der S3-Bucket, den Sie für den Trail angeben, muss über eine Bucket-Richtlinie verfügen, die CloudTrail das Schreiben von Protokolldateien in ihn ermöglicht. Weitere Informationen finden Sie unter [Amazon S3 S3-Bucket-Richtlinie für CloudTrail](#).

1. Erstellen Sie einen Trail:

```
aws cloudtrail create-trail \  
  --name cloudwatch-logs-api-key-audit \  
  --s3-bucket-name my-cloudtrail-bucket \  
  --region us-east-1
```

2. Konfigurieren Sie erweiterte Event-Selektoren, um Logs zu CloudWatch protokollieren, Gruppenautorisierungsereignisse zu protokollieren:

```
aws cloudtrail put-event-selectors \  
  --region us-east-1 \  

```

```
--trail-name cloudwatch-logs-api-key-audit \  
--advanced-event-selectors ' [{  
    "Name": "CloudWatch Logs API key authorization events",  
    "FieldSelectors": [  
        { "Field": "eventCategory", "Equals": ["Data"] },  
        { "Field": "resources.type", "Equals":  
["AWS::Logs::LogGroupAuthorization"] }  
    ]  
}]'
```

3. Starten Sie die Trailprotokollierung:

```
aws cloudtrail start-logging \  
--name cloudwatch-logs-api-key-audit \  
--region us-east-1
```

Senden von Protokollen über den OTLP-Endpunkt (OpenTelemetry Logs)

Der OpenTelemetry Logs-Endpunkt (/v1/logs) akzeptiert OpenTelemetry Protokolldaten des Protokolls (OTLP) entweder in JSON- oder Protobuf-Kodierung. Ausführliche Informationen zum OTLP-Endpunkt, einschließlich Konfiguration und Verwendung, finden [Sie unter Metriken und Traces an mit senden](#). CloudWatch OpenTelemetry

Wenn Sie die Bearer-Token-Authentifizierung verwenden, schließen Sie die Einrichtungsschritte unter ab, [the section called “Bearer-Token-Authentifizierung”](#) bevor Sie fortfahren.

Anforderungsformat

- Methode: POST
- Inhaltstyp: oder application/json application/x-protobuf
- Protokollgruppe: nur x-aws-log-group Header (Abfrageparameter nicht unterstützt)
- Protokollstream: x-aws-log-stream Header

Beispielanforderung

```
curl -X POST "https://logs.<region>.amazonaws.com/v1/logs" \  
-H "Authorization: Bearer ACWL<token>" \  

```

```
-H "Content-Type: application/json" \  
-H "x-aws-log-group: MyLogGroup" \  
-H "x-aws-log-stream: MyLogStream" \  
-d '{  
  "resourceLogs": [  
    {  
      "resource": {  
        "attributes": [  
          {  
            "key": "service.name",  
            "value": { "stringValue": "my-service" }  
          }  
        ]  
      },  
      "scopeLogs": [  
        {  
          "scope": {  
            "name": "my-library",  
            "version": "1.0.0"  
          },  
          "logRecords": [  
            {  
              "timeUnixNano": "1741900000000000000",  
              "severityNumber": 9,  
              "severityText": "INFO",  
              "body": {  
                "stringValue": "User logged in successfully"  
              },  
              "attributes": [  
                {  
                  "key": "user.id",  
                  "value": { "stringValue": "12345" }  
                }  
              ]  
            }  
          ]  
        }  
      ]  
    }  
  ]  
}'
```

Antworten

Erfolgreich (alle Ereignisse wurden akzeptiert):

```
HTTP 200 OK
{}
```

Teilweise erfolgreich (einige Ereignisse wurden abgelehnt):

```
{
  "partialSuccess": {
    "rejectedLogRecords": 5,
    "errorMessage": "{\\"tooOldLogEventCount\\": 3, \\"tooNewLogEventCount\\": 1, \\"expiredLogEventCount\\": 1}"
  }
}
```

Wenn die Anfrage Content-Type lautet `application/x-protobuf`, wird die Antwort als serialisierte `ExportLogsServiceResponse` Protobuf-Nachricht mit denselben Feldern zurückgegeben.

OTLP-spezifisches Verhalten

Die folgenden Verhaltensweisen sind spezifisch für den OTLP-Endpunkt und nicht auf den anderen HTTP-Ingestion-Endpunkten vorhanden:

- **Retry-After-Header** — In den Antworten 503 und 429 enthalten und gibt an, wann der Client es erneut versuchen sollte.

Senden von Protokollen über den HLC-Endpunkt (HLC Logs)

Der HLC Logs-Endpunkt (`/services/collector/event`) basiert auf dem HTTP Log Collector (HLC) -Format.

Wenn Sie die Bearer-Token-Authentifizierung verwenden, schließen Sie die Einrichtungsschritte unter ab, bevor Sie fortfahren. [the section called “Bearer-Token-Authentifizierung”](#)

Eingabemodi

Jedes Ereignis ist ein JSON-Objekt mit einem erforderlichen `"event"` Feld. Optionale Metadatenfelder: `"time"` `"host"`, `"source"`, `"sourcetype"`, `"index"`.

Einzelnes Ereignis:

```
{"event":"Hello world!","time":1486683865.0}
```

JSON-Array von Ereignissen:

```
[  
  {"event":"msg1","time":1486683865.0},  
  {"event":"msg2","time":1486683866.0}  
]
```

Verkettete/gebündelte Ereignisse (kein Array-Wrapper):

```
{"event":"msg1","time":1486683865.0}{ "event":"msg2","time":1486683866.0}
```

Feld „Ereignis“ (erforderlich)

Das "event" Feld ist ein Pflichtfeld. Sein Wert kann ein beliebiger JSON-Typ sein:

```
{"event":"a string message"}  
{"event":{"message":"structured data","severity":"INFO"}}  
{"event":42}  
{"event":true}
```

Objekte ohne "event" Feld werden stillschweigend übersprungen:

```
{"message":"this is skipped – no event field"}
```

Zeitfeld (optional)

Das "time" Feld wird in Epochensekunden (nicht in Millisekunden) angegeben, mit optionaler Dezimalzahl für eine Genauigkeit unter einer Sekunde.

Format	Beispiel	Interpretiert als
Gleitkommazahl	"time":1486683865.500	1486683865500 ms
Ganzzahl	"time":1486683865	1486683865000 ms

Format	Beispiel	Interpretiert als
Zeichenfolge (Float)	"time":"1486683865.500"	1486683865500 ms
Zeichenfolge (Ganzzahl)	"time":"1486683865"	1486683865000 ms
Fehlen	(kein Zeitfeld)	Aktuelle Uhrzeit des Servers
Ungültig	"time":"invalid"	Aktuelle Uhrzeit des Servers

Content-Type

Nur `application/json` wird akzeptiert.

Akzeptierte JSON-Werttypen

Typ der obersten Ebene	Behavior
Objekt mit "event"	Accepted (Akzeptiert)
Objekt ohne "event"	Übersprungen
Array von -Objekten.	Jedes Element wird einzeln verarbeitet
Verkettete Objekte	Jedes Objekt wird einzeln verarbeitet
Primitiv (Zeichenfolge, Zahl, Boolean, Null)	Übersprungen

Endpunkt-Format

Die URL des HLC-Endpunkts folgt diesem Format:

```
https://logs.<region>.amazonaws.com/services/collector/event?
logGroup=<name>&logStream=<name>[&entityName=<name>&entityEnvironment=<environment>]
```

Erforderliche Parameter:

- `<region>`— AWS Region (zum Beispiel `us-east-1`, `eu-west-1`)
- `logGroup`— URL-codierter Name der Protokollgruppe
- `logStream`— URL-codierter Name des Protokolldatenstroms

Optionale Parameter:

Sie können Ihre Protokollereignisse optional einer Service Entität zuordnen, indem Sie die folgenden Abfrageparameter angeben. Da es sich bei den über den HLC-Endpunkt gesendeten Protokollen um benutzerdefinierte Telemetrie handelt, werden sie nicht automatisch einer Entität zugeordnet. Durch die Angabe dieser Parameter erstellt CloudWatch Logs eine Entität mit der `KeyAttributes.Type` Einstellung auf `Service` und ordnet sie Ihren Protokollereignissen zu. Auf diese Weise kann die zugehörige CloudWatch Explore-Funktion diese Protokolle mit anderen Telemetriedaten (Metriken, Traces und Logs) desselben Dienstes korrelieren, was die Fehlerbehebung und Überwachung Ihrer Anwendungen bei verschiedenen Signaltypen erleichtert. Weitere Informationen zu Entitäten und verwandter Telemetrie finden Sie unter [Hinzufügen verwandter Informationen zur](#) benutzerdefinierten Telemetrie.

- `entityName`— Der Name der Dienstentität, die den Protokollereignissen zugeordnet werden soll. Dieser Wert wird als Entität gespeichert `KeyAttributes.Name` (z. B. `my-application` oder `api.myservice.com`).
- `entityEnvironment`— Die Umgebung, in der der Dienst gehostet wird oder zu der er gehört. Dieser Wert wird als Entität gespeichert `KeyAttributes.Environment` (z. B. `production`, `ec2:default`, oder `eks:my-cluster/default`).

Anforderungsformat

Senden Sie Protokolle mithilfe von HTTP POST mit den folgenden Headern und dem folgenden Hauptteil:

Kopfzeilen:

- `Authorization: Bearer <your-bearer-token>`
- `Content-Type: application/json`

Körperformat:

Der Hauptteil der Anfrage sollte im JSON-Format mit einer Reihe von Ereignissen vorliegen:

```
{
  "event": [
    {
      "time": 1730141374.001,
      "event": "Application started successfully",
      "host": "web-server-1",
      "source": "application.log",
      "severity": "info"
    },
    {
      "time": 1730141374.457,
      "event": "User login successful",
      "host": "web-server-1",
      "source": "auth.log",
      "user": "john.doe"
    }
  ]
}
```

Feldbeschreibungen:

- **time**— Zeitstempel der Unix-Epoche in Sekunden, mit optionaler Dezimalzahl für eine Genauigkeit unter einer Sekunde (optional)
- **event**— Die Protokollnachricht oder die Ereignisdaten (erforderlich)
- **host**— Hostname oder Kennung der Quelle (optional)
- **source**— Kennung der Protokollquelle (optional)

Zusätzliche benutzerdefinierte Felder können nach Bedarf hinzugefügt werden.

Beispielanforderung

```
curl -X POST "https://logs.<region>.amazonaws.com/services/collector/event?
logGroup=MyLogGroup&logStream=MyStream" \
-H "Authorization: Bearer ACWL<token>" \
-H "Content-Type: application/json" \
-d '{"event":{"message":"User logged
in","user_id":"u-123"},"time":1486683865.0,"host":"web-01","source":"auth-service"}'
```

Best Practices

Ereignisse stapeln

Für mehr Leistung und Effizienz:

- Wenn möglich, mehrere Ereignisse in einer einzigen Anfrage bündeln
- Empfohlene Batchgröße: 10—100 Ereignisse pro Anfrage
- Maximale Anforderungsgröße: 1 MB

Fehlerbehandlung

Implementieren Sie die richtige Fehlerbehandlung in Ihrer Anwendung. Allgemeine HTTP-Statuscodes:

- 200 OK— Protokolle wurden erfolgreich aufgenommen
- 400 Bad Request— Ungültiges Anforderungsformat oder ungültige Parameter
- 401 Unauthorized— Ungültiges oder abgelaufenes Inhaber-Token
- 403 Forbidden— Unzureichende Berechtigungen
- 404 Not Found— Protokollgruppe oder Stream existiert nicht
- 429 Too Many Requests— Ratenlimit überschritten
- 500 Internal Server Error— Servicefehler (erneuter Versuch mit exponentiellem Backoff)

Einschränkungen

- Maximale Ereignisgröße: 256 KB pro Ereignis
- Maximale Anforderungsgröße: 1 MB
- Maximale Anzahl von Ereignissen pro Anfrage: 10.000
- Die Namen der Protokollgruppen müssen den Benennungskonventionen für CloudWatch Protokolle entsprechen
- Wenn die Bearer-Token-Authentifizierung verwendet wird, muss die Bearer-Token-Authentifizierung für die Protokollgruppe aktiviert sein.

Senden von Protokollen mithilfe des NDJSON-Endpunkts (ND-JSON-Protokolle)

Der Endpunkt ND-JSON Logs (/ingest/bulk) akzeptiert Protokolle im [NDJSON-Format](#) (Newline Delimited JSON). Jede Zeile enthält genau einen JSON-Wert, getrennt durch Zeilenumbruchzeichen.

Wenn Sie die Bearer-Token-Authentifizierung verwenden, schließen Sie die Einrichtungsschritte unter ab, [the section called "Bearer-Token-Authentifizierung"](#) bevor Sie fortfahren.

Anforderungsformat

Senden Sie einen JSON-Wert pro Zeile, getrennt durch \n (LF) oder \r\n (CRLF). Leere Zeilen werden stillschweigend ignoriert.

```
{"timestamp":1771007942000,"message":"event one","level":"INFO"}
{"timestamp":1771007943000,"message":"event two","level":"ERROR"}
{"timestamp":1771007944000,"message":"event three","level":"DEBUG"}
```

application/json Sowohl als auch application/x-ndjson werden als Content-Type akzeptiert.

Akzeptierte JSON-Werttypen

Gemäß der NDJSON-Spezifikation (RFC 8259) wird jeder gültige JSON-Wert in jeder Zeile akzeptiert.

JSON-Objekte (am häufigsten):

```
{"timestamp":1771007942000,"message":"User logged in","service":"auth"}
{"timestamp":1771007943000,"error":"Connection timeout","service":"api"}
```

JSON-Arrays (auf einzelne Ereignisse reduziert):

```
[{"timestamp":1000,"message":"a"}, {"timestamp":2000,"message":"b"}]
```

Diese einzelne Zeile erzeugt 2 Ereignisse. Jedes Array-Element wird zu einem separaten Protokollereignis.

Primitive Werte:

```
"a plain string log message"
42
true
null
```

Jedes Primitiv wird zu einem eigenen Ereignis mit dem aktuellen Zeitstempel des Servers.

Gemischte Typen:

```
{"timestamp":1771007942000,"message":"structured event"}
"unstructured string message"
42
{"timestamp":1771007943000,"error":"something failed"}
```

Alle 4 Zeilen werden als gültige Ereignisse akzeptiert.

Inhalt der Zeile	Behavior
JSON-Objekt	Akzeptiert, Zeitstempel extrahiert, falls vorhanden
JSON-Array	Reduziert — jedes Element wird zu einem separaten Ereignis
Leeres Array []	Akzeptiert, erzeugt 0 Ereignisse
JSON-Zeichenfolge	Als Ereignisnachricht akzeptiert
JSON-Nummer	Als Ereignisnachricht akzeptiert
Boolescher JSON-Wert	Als Ereignisnachricht akzeptiert
JSON ist null	Als Ereignisnachricht akzeptiert

Inhalt der Zeile	Behavior
Ungültige JSON	Übersprungen (gezählt, Verarbeitung wird fortgesetzt)
Leere Zeile	Ignoriert (wird nicht als übersprungen gezählt)

Zeitstempel-Feld

Das "timestamp" Feld ist in Epochen-Millisekunden (nicht Sekunden) angegeben.

Format	Beispiel	Interpretiert als
Numerisch (Millis)	"timestamp":1771007942000	1771007942000 ms
Fehlen	(kein Zeitstempelfeld)	Aktuelle Uhrzeit des Servers
Nicht numerisch	"timestamp":"invalid"	Aktuelle Uhrzeit des Servers
Zeile, die kein Objekt ist	"hello", 42, true	Aktuelle Uhrzeit des Servers

Ungültige Zeilen

Zeilen, die kein gültiges JSON-Format sind, werden automatisch übersprungen und gezählt. Die Verarbeitung wird mit der nächsten Zeile fortgesetzt.

```
{"message":"valid event"}
this is not valid json
{"message":"another valid event"}
```

Ergebnis: 2 Ereignisse wurden aufgenommen, 1 wurde übersprungen. Gibt HTTP 200 zurück.

Wenn alle Zeilen ungültig sind, wird mit zurückgegeben HTTP 400. "All events were invalid"

Beispielanforderung

```
curl -X POST "https://logs.<region>.amazonaws.com/ingest/bulk?
logGroup=MyLogGroup&logStream=MyStream" \
-H "Authorization: Bearer ACWL<token>" \
-H "Content-Type: application/x-ndjson" \
-d '{"timestamp":1771007942000,"message":"User logged in","level":"INFO"}
{"timestamp":1771007943000,"message":"Query took 42ms","level":"DEBUG"}
{"timestamp":1771007944000,"error":"Connection refused","level":"ERROR"}'
```

Antworten

Erfolgreich (alle Ereignisse wurden akzeptiert):

```
HTTP 200 OK
{}
```

Teilweise erfolgreich (einige Ereignisse wurden abgelehnt):

```
{
  "partialSuccess": {
    "rejectedLogRecords": 5,
    "errorMessage": "{\"tooOldLogEventCount\": 3, \"tooNewLogEventCount\": 1,
    \"expiredLogEventCount\": 1}"
  }
}
```

Das `rejectedLogRecords` Feld gibt die Gesamtzahl der abgelehnten Ereignisse an. Das `errorMessage` Feld enthält eine JSON-kodierte Aufschlüsselung nach Ablehnungsgründen:

- `tooOldLogEventCount`— Ereignisse, deren Zeitstempel älter als der Aufbewahrungszeitraum sind
- `tooNewLogEventCount`— Ereignisse mit Zeitstempeln, die zu weit in der future liegen
- `expiredLogEventCount`— Ereignisse, die während der Verarbeitung abgelaufen sind

Best Practices

Ereignisse werden gebündelt

Für mehr Leistung und Effizienz:

- Wenn möglich, mehrere Ereignisse in einer einzigen Anfrage bündeln
- Empfohlene Batchgröße: 10—100 Ereignisse pro Anfrage
- Maximale Anforderungsgröße: 1 MB

Fehlerbehandlung

Implementieren Sie die richtige Fehlerbehandlung in Ihrer Anwendung. Allgemeine HTTP-Statuscodes:

- 200 OK— Protokolle wurden erfolgreich aufgenommen
- 400 Bad Request— Ungültiges Anforderungsformat oder ungültige Parameter
- 401 Unauthorized— Ungültiges oder abgelaufenes Trägertoken
- 403 Forbidden— Unzureichende Berechtigungen
- 404 Not Found— Protokollgruppe oder Stream existiert nicht
- 429 Too Many Requests— Ratenlimit überschritten
- 500 Internal Server Error— Servicefehler (erneuter Versuch mit exponentiellem Backoff)

Einschränkungen

- Maximale Ereignisgröße: 256 KB pro Ereignis
- Maximale Anforderungsgröße: 1 MB
- Maximale Anzahl von Ereignissen pro Anfrage: 10.000
- Die Namen der Protokollgruppen müssen den Benennungskonventionen für CloudWatch Protokolle entsprechen
- Wenn die Bearer-Token-Authentifizierung verwendet wird, muss die Bearer-Token-Authentifizierung für die Protokollgruppe aktiviert sein.

Senden von Protokollen mithilfe des strukturierten JSON-Endpunkts (Strukturierte JSON-Protokolle)

Der Endpunkt Structured JSON Logs (/ingest/json) akzeptiert Standard-JSON — entweder ein einzelnes JSON-Objekt oder ein JSON-Array von Objekten. Dieser Endpunkt ist für strukturierte Protokolldaten konzipiert, bei denen jedes Ereignis ein JSON-Objekt ist.

Wenn Sie die Bearer-Token-Authentifizierung verwenden, schließen Sie die Einrichtungsschritte unter ab, [the section called “Bearer-Token-Authentifizierung”](#) bevor Sie fortfahren.

Anforderungsformat

application/json Wird nur als Content-Type akzeptiert.

Einzelnes JSON-Objekt:

```
{"timestamp":1771007942000,"message":"single event","level":"INFO"}
```

JSON-Array von Objekten:

```
[{"timestamp":1771007942000,"message":"event one","level":"INFO"}, {"timestamp":1771007943000,"message":"event two","level":"ERROR"}]
```

Akzeptierte JSON-Werttypen

Dieser Endpunkt ist streng — nur JSON-Objekte werden als Ereignisse akzeptiert.

Input	Behavior
Einzelnes JSON-Objekt	Als ein Ereignis akzeptiert
JSON-Array von Objekten	Jedes Objekt wird zu einem separaten Ereignis
Leeres Array []	Akzeptiert, erzeugt 0 Ereignisse

Input	Behavior
Kein Objekt im Array (Zeichenfolge, Zahl usw.)	Übersprungen
Primitiv der obersten Ebene ("hello",) 42	Übersprungen
Verkettete Objekte {...}{...}	Nur das erste Objekt wurde analysiert

Beispiel — Array mit gemischten Typen:

```
[
  {"timestamp":1771007942000,"message":"valid object"},
  "just a string",
  42,
  {"timestamp":1771007943000,"message":"another valid object"}
]
```

Ergebnis: 2 Ereignisse wurden aufgenommen (die Objekte), 2 übersprungen (die Zeichenfolge und die Zahl).

Feld „Zeitstempel“

Das "timestamp" Feld wird in Epochen-Millisekunden angegeben, genau wie der NDJSON-Endpunkt.

Format	Beispiel	Interpretiert als
Numerisch (Millis)	"timestamp":1771007942000	1771007942000 ms
Fehlen	(kein Zeitstempelfeld)	Aktuelle Uhrzeit des Servers
Nicht numerisch	"timestamp":"invalid"	Aktuelle Uhrzeit des Servers

Beispielanforderung

```
curl -X POST "https://logs.<region>.amazonaws.com/ingest/json?
logGroup=MyLogGroup&logStream=MyStream" \
-H "Authorization: Bearer ACWL<token>" \
-H "Content-Type: application/json" \
-d '[{"timestamp":1771007942000,"message":"User logged in","user_id":"u-123"},
{"timestamp":1771007943000,"message":"Order placed","order_id":"o-456"}]'
```

Antworten

Erfolgreich (alle Ereignisse wurden akzeptiert):

```
HTTP 200 OK
{}
```

Teilweise erfolgreich (einige Ereignisse wurden abgelehnt):

```
{
  "partialSuccess": {
    "rejectedLogRecords": 5,
    "errorMessage": "{\"tooOldLogEventCount\": 3, \"tooNewLogEventCount\": 1,
    \"expiredLogEventCount\": 1}"
  }
}
```

Das `rejectedLogRecords` Feld gibt die Gesamtzahl der abgelehnten Ereignisse an. Das `errorMessage` Feld enthält eine JSON-kodierte Aufschlüsselung nach Ablehnungsgründen:

- `tooOldLogEventCount`— Ereignisse, deren Zeitstempel älter als der Aufbewahrungszeitraum sind
- `tooNewLogEventCount`— Ereignisse mit Zeitstempeln, die zu weit in der future liegen
- `expiredLogEventCount`— Ereignisse, die während der Verarbeitung abgelaufen sind

Best Practices

Ereignisse werden gebündelt

Für mehr Leistung und Effizienz:

- Wenn möglich, mehrere Ereignisse in einer einzigen Anfrage bündeln
- Empfohlene Batchgröße: 10—100 Ereignisse pro Anfrage
- Maximale Anforderungsgröße: 1 MB

Fehlerbehandlung

Implementieren Sie die richtige Fehlerbehandlung in Ihrer Anwendung. Allgemeine HTTP-Statuscodes:

- 200 OK— Protokolle wurden erfolgreich aufgenommen
- 400 Bad Request— Ungültiges Anforderungsformat oder ungültige Parameter
- 401 Unauthorized— Ungültiges oder abgelaufenes Trägertoken
- 403 Forbidden— Unzureichende Berechtigungen
- 404 Not Found— Protokollgruppe oder Stream existiert nicht
- 429 Too Many Requests— Ratenlimit überschritten
- 500 Internal Server Error— Servicefehler (erneuter Versuch mit exponentiellem Backoff)

Einschränkungen

- Maximale Ereignisgröße: 256 KB pro Ereignis
- Maximale Anforderungsgröße: 1 MB
- Maximale Anzahl von Ereignissen pro Anfrage: 10.000
- Die Namen der Protokollgruppen müssen den Benennungskonventionen für CloudWatch Protokolle entsprechen
- Wenn die Bearer-Token-Authentifizierung verwendet wird, muss die Bearer-Token-Authentifizierung für die Protokollgruppe aktiviert sein.

Vergleich der Endpunkte für die HTTP-Erfassung

Feature	HLC-Protokolle	ND-JSON-Protokolle	Strukturierte JSON-Protokolle	OpenTelemetry Logs
Pfad	/services/collector/event	/ingest/bulk	/ingest/json	/v1/logs
Content-Type	application/json	application/json oder application/x-ndjson	application/json	application/json oder application/x-protobuf
Feld „Zeitstempel“	"time"(Sekunden)	"timestamp" (Millisekunden)	"timestamp" (Millisekunden)	"timeUnixNano" (Nanosekunden)
Pflichtfelder	"event"	Keine	Keine	OTLP-Struktur() "resourceLogs"
Teilweise erfolgreiche Antwort	Nein	Ja	Ja	Ja
Unterstützung von Abfrageparametern	Ja	Ja	Ja	Nein (nur Kopfzeilen)
Metadaten der Entität	Ja	Ja	Ja	Nein
Akzeptiert Primitive	Nein	Ja	Nein	Nein
Zeilenbasiertes Parsen	Nein	Ja	Nein	Nein
Protobuf-Unterstützung	Nein	Nein	Nein	Ja

Feature	HLC-Protokolle	ND-JSON-Protokolle	Strukturierte JSON-Protokolle	OpenTelemetry Logs
Header „Nach einem erneuten Versuch“	Nein	Nein	Nein	Ja

Einen Endpunkt auswählen

- Verwenden Sie das HLC-Format? Verwenden Sie HLC-Protokolle. Ihre vorhandenen HLC-Payloads funktionieren mit minimalen Änderungen.
- Protokolle streamen? line-by-line Verwenden Sie ND-JSON-Protokolle. Am besten für Protokoll-Pipelines geeignet, die ein Ereignis pro Zeile ausgeben. Am flexibelsten — akzeptiert jeden JSON-Wertetyp.
- Strukturierte JSON-Nutzlasten senden? Verwenden Sie strukturierte JSON-Protokolle. Am besten für Anwendungen geeignet, die wohlgeformte JSON-Objekte oder -Arrays erzeugen.
- Benutzen Sie es bereits? OpenTelemetry Verwenden Sie OpenTelemetry Logs. Akzeptiert das OTLP-JSON- oder Protobuf-Format und unterstützt teilweise erfolgreiche Antworten mit Wiederholungssemantik.

Verwenden von CloudWatch Logs mit einem AWS SDK

AWS Software Development Kits (SDKs) sind für viele beliebte Programmiersprachen verfügbar. Jedes SDK bietet eine API, Codebeispiele und Dokumentation, die es Entwicklern erleichtern, Anwendungen in ihrer bevorzugten Sprache zu erstellen.

SDK-Dokumentation	Codebeispiele
AWS SDK für C++	AWS SDK für C++ Code-Beispiele
AWS CLI	AWS CLI Code-Beispiele
AWS SDK für Go	AWS SDK für Go Code-Beispiele
AWS SDK für Java	AWS SDK für Java Code-Beispiele
AWS SDK für JavaScript	AWS SDK für JavaScript Code-Beispiele
AWS SDK für Kotlin	AWS SDK für Kotlin Code-Beispiele
AWS SDK für .NET	AWS SDK für .NET Code-Beispiele
AWS SDK für PHP	AWS SDK für PHP Code-Beispiele
AWS -Tools für PowerShell	AWS -Tools für PowerShell Code-Beispiele
AWS SDK für Python (Boto3)	AWS SDK für Python (Boto3) Code-Beispiele
AWS SDK für Ruby	AWS SDK für Ruby Code-Beispiele
AWS SDK für Rust	AWS SDK für Rust Code-Beispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Code-Beispiele
AWS SDK für Swift	AWS SDK für Swift Code-Beispiele

Spezifische Beispiele für CloudWatch Logs finden Sie unter [Codebeispiele für CloudWatch Logs mit AWS SDKs](#).

 Beispiel für die Verfügbarkeit

Sie können nicht finden, was Sie brauchen? Fordern Sie ein Codebeispiel an, indem Sie unten den Link [Provide feedback \(Feedback geben\)](#) auswählen.

Verwaltung von Protokollen

CloudWatch Logs bietet erweiterte Protokollverwaltungsfunktionen, mit denen Sie Ihre Protokolldaten effektiver organisieren, transformieren und analysieren können. Zu diesen Funktionen gehören die [konto- und regionsübergreifende Datenzentralisierung](#), [automatische Datenerkennung und Schemaverwaltung](#), [Protokolltransformation bei der Erfassung](#) und [erweiterte Analysen mit Facetten](#) für die interaktive Protokollerkundung.

Themen

- [Erkennung und Verwaltung von Datenquellen](#)
- [Funktionen, die durch Datenquellen ermöglicht werden](#)
- [Unterstützte Datenquellen](#)

Erkennung und Verwaltung von Datenquellen

CloudWatch Logs erkennt und kategorisiert Ihre Protokolldaten automatisch nach Datenquelle und Typ, sodass Sie Ihre Protokolle leichter verstehen und in großem Umfang verwalten können. Diese Funktion ermöglicht die Schemaerkennung AWS für verkaufte Quellen wie Amazon VPC Flow Logs und Route 53 sowie Sicherheitstools von Drittanbietern. CloudTrail

Die Logs Management-Konsole bietet eine allgemeine Ansicht Ihrer Protokolle, die nach Datenquelle und Typ geordnet sind, und nicht nur nach Protokollgruppen. Diese Organisation hilft Ihnen dabei:

- Zeigen Sie Protokolle an, die nach AWS Diensten, Quellen von Drittanbietern (wie Okta oder CrowdStrike) und benutzerdefinierten Quellen kategorisiert sind
- Machen Sie sich automatisch mit dem Schema und der Struktur Ihrer Protokolldaten vertraut
- Erstellen Sie Feldindexrichtlinien auf der Grundlage erkannter Schemafelder
- Verwalten Sie Protokolle effizienter über verschiedene Datenquellen hinweg
- Logs nach verschiedenen Datenquellen abfragen

Wenn Sie die [CloudWatch Protokollprotokollierung für unterstützte AWS Dienste aktivieren](#), wendet CloudWatch Logs automatisch das entsprechende Schema auf Ihre Protokolle an. Diese automatische Schemaanwendung trägt zur Wahrung der Konsistenz bei und bietet sofortige Einblicke in Ihre Protokollstruktur.

Was sind CloudWatch Logs-Datenquellen?

CloudWatch Logs Data Sources ist eine Funktion, die eine neue Möglichkeit bietet, Ihre Protokolldaten anhand der Quelle zu organisieren und zu kategorisieren, die die Protokolle generiert. Während CloudWatch Logs traditionell Protokollgruppen zum Organisieren von Protokollen verwendet, bietet Data Sources eine zusätzliche Organisationsebene, bei der Protokolle nach ihrem ursprünglichen Dienst und Protokolltyp gruppiert werden.

Wie funktionieren Datenquellen

Datenquellen bieten eine servicebasierte Protokollorganisation und eine vereinfachte Erkennung in Ihrer gesamten AWS Infrastruktur. Sie können problemlos nach Protokollen bestimmter Dienste suchen und nach Protokolltyp filtern, ohne dass Sie die Namen oder Strukturen der einzelnen Protokollgruppen kennen müssen.

Bei Quellen von Drittanbietern und optional bei Anwendungsprotokollquellen arbeiten Datenquellen mit CloudWatch Pipelines zusammen, um Ihre Protokolle zu kategorisieren. Wenn Sie eine Pipeline für die Aufnahme und Transformation von Protokollen konfigurieren, geben Sie den Namen und den Typ der Datenquelle an. CloudWatch Logs kategorisiert dann automatisch alle Protokolle, die von der Pipeline verarbeitet werden. Weitere Informationen finden Sie unter [CloudWatch Pipelines](#) im CloudWatch Amazon-Benutzerhandbuch.

Datenquellen kategorisieren Protokolle anhand von zwei Schlüsselkennungen:

- **Datenquellenname:** Der AWS Service, die Drittanbieterquelle oder die Anwendung, die die Protokolle generiert (z. B. Route 53, Amazon VPC CloudTrail, Okta SSO oder CrowdStrike Falcon).
- **Datenquellentyp:** Der spezifische Protokolltyp, der von diesem Dienst generiert wurde.

Ein Schema definiert die Struktur von Protokolldaten, einschließlich der vorhandenen Felder und der Organisation der Informationen. Eine einzelne Datenquelle kann mehrere Arten von Protokollen mit unterschiedlichen Schemas und Zwecken erzeugen. Bei der AWS CloudTrail Datenquelle gibt es beispielsweise zwei Typen: Verwaltungsereignisse (die Vorgänge auf der Kontrollebene wie das Erstellen oder Löschen von Ressourcen verfolgen) und Datenereignisse (die Datenebenenvorgänge wie S3-Objektzugriffe verfolgen). Jeder Typ hat ein anderes Schema, da er unterschiedliche Arten von Informationen erfasst.

Erste Schritte

CloudWatch Logs kategorisiert Ihre Logs anhand ihrer Herkunft in Datenquellen. Die Methode hängt von der Art der Protokolle ab, mit denen Sie arbeiten:

AWS-Service Protokolle

AWS-Services Die [unterstützten](#) Protokolle werden automatisch nach Datenquellen gruppiert, ohne dass eine Konfiguration erforderlich ist. CloudWatch Logs erkennt diese Protokolle und wendet den entsprechenden Datenquellennamen und -typ auf der Grundlage des ursprünglichen Dienstes an.

Third-party Logs

Third-party Protokolle erfordern Pipelines für die Kategorisierung von Datenquellen. Wenn Sie eine Pipeline so konfigurieren, dass sie Protokolle aus unterstützten Drittanbieterquellen wie Microsoft Office 365, Okta oder Palo Alto Networks aufnimmt CrowdStrike, geben Sie den [Namen und den Typ der Datenquelle](#) in der Pipeline-Konfiguration an. CloudWatch Logs kategorisiert automatisch alle Protokolle, die von der Pipeline verarbeitet werden, anhand dieser Kennungen.

Pipelines können optional Protokolle von Drittanbietern in das Open Cybersecurity Schema Framework (OCSF) -Format umwandeln, um eine standardisierte Analyse von Sicherheitsereignissen zu ermöglichen. Wenn die OCSF-Transformation aktiviert ist, werden Name und Typ der Datenquelle automatisch auf der Grundlage der OCSF-Schemazuordnung bestimmt. Ohne OCSF-Transformation geben Sie den Namen und den Typ der Datenquelle in der Pipeline-Konfiguration an.

Anwendungsprotokolle

Benutzerdefinierte Anwendungsprotokolle können Sie mit einer der folgenden Methoden nach Datenquelle kategorisieren:

- **Protokollgruppen-Tags** — Fügen Sie mithilfe der Schlüssel Tags zu Ihren Protokollgruppen hinzu `cw:datasource:name` und `cw:datasource:type` geben Sie jeweils den Namen und den Typ der Datenquelle für alle in der Protokollgruppe aufgenommenen Protokolle an. Tag-Werte können bis zu 64 Zeichen lang sein und dürfen nur Kleinbuchstaben, Zahlen und Unterstriche enthalten. Sie müssen entweder mit einem Buchstaben oder einer Zahl beginnen und dürfen keine doppelten Unterstriche (__) enthalten.
- **Pipeline-Konfiguration** — Konfigurieren Sie Datenquelleninformationen mithilfe von Protokollverarbeitungspipelines, wenn Sie Ihre Anwendungsprotokolle aufnehmen.

 Note

Datenquellennamen dürfen nicht mit „aws“ oder „amazon“ beginnen, um Konflikte mit AWS Serviceprotokollen zu vermeiden.

Systemfelder

CloudWatch Logs fügt Logs automatisch drei Systemfelder zu Protokollen hinzu, die nach Datenquelle kategorisiert sind. Diese Felder dienen als Standardfacetten:

- @data_source_name- Enthält den Namen der Datenquelle oder „Unbekannt“, falls nicht bestimmt
- @data_source_type- Enthält den Typ der Datenquelle oder „Unbekannt“, falls nicht bestimmt
- @data_format- Gibt das Format der Protokolldaten an

Wenn der Name oder Typ der Datenquelle nicht bestimmt werden kann, werden diese Felder auf „Unbekannt“ gesetzt. Datenquellen mit „Unbekannten“ Werten sind weiterhin in Facetten und in der Datenquellentabelle unter „Protokollverwaltung“ in der Konsole sichtbar, sodass Sie nicht kategorisierte Protokolle und deren Herkunft identifizieren können.

Das @data_format Feld kann einen der folgenden Werte enthalten:

- Default- Logs, die ohne Änderung aufgenommen wurden.
- Custom- Protokolle, die über Pipelineprozessoren verarbeitet wurden, oder Protokolle, die in eine Protokollgruppe mit name/type Datenquellen-Tags aufgenommen wurden.
- OCSF-<version>- Protokolle, die mit OCSF-Prozessoren (Open Cybersecurity Schema Framework) in Pipelines verarbeitet wurden.
- AWS-OTEL-LOG-V<version>- OpenTelemetry Protokolle, die über den OTLP-Endpunkt aufgenommen wurden CloudWatch .
- AWS-OTEL-TRACE-V<version>- OpenTelemetry Traces, die über den OTLP-Endpunkt aufgenommen wurden CloudWatch .

Mit diesen Systemfeldern können Sie Ihre Logs anhand ihrer Quelle und ihres Formats filtern und abfragen, was die Arbeit mit Logs unterschiedlicher Herkunft und Verarbeitungspipelines erleichtert.

Zugreifen auf Datenquellen

Konsole

In der CloudWatch Protokollkonsole verwenden Sie die Registerkarte Protokollverwaltung, um auf Ihre Datenquellen zuzugreifen. CloudWatch Logs konsolidiert Ihre Protokolldaten automatisch nach Datenquellen und -typen und entdeckt dabei kontinuierlich neu aufgenommene Daten. In der Datenquellenliste können Sie Pipelines erstellen und Feldindizes und Facetten definieren.

AWS CLI

Verwenden Sie den folgenden Befehl, um verschiedene Datenquellen und Protokolltypen in Ihrem Konto aufzulisten:

```
aws logs list-aggregate-log-group-summaries --group-by DATA_SOURCE_NAME_AND_TYPE
```

Beziehung zu Protokollgruppen

Datenquellen ergänzen Protokollgruppen, anstatt sie zu ersetzen. Ihre Protokolle werden weiterhin wie bisher in Protokollgruppen gespeichert, aber jetzt werden sie auch automatisch mit Datenquelleninformationen gekennzeichnet. Diese duale Organisation ermöglicht Ihnen:

- Verwenden Sie Protokollgruppen für differenzierte Zugriffskontroll- und Aufbewahrungsrichtlinien
- Verwenden Sie Datenquellen für die servicebasierte Erkennung und Analyse von Protokollen
- Fragen Sie Protokolle je nach Bedarf mit einer der beiden Organisationsmethoden ab

Datenquellen erleichtern die Arbeit mit Protokollen in großem Umfang, indem sie eine serviceorientierte Ansicht Ihrer Protokolldaten in Ihrer gesamten AWS Infrastruktur bieten.

Funktionen, die durch Datenquellen ermöglicht werden

Datenquellen ermöglichen erweiterte Funktionen zur Protokollverarbeitung und -analyse durch Felderkennung und konsistente Datenstrukturen.

- Facetten: Facetten sind indizierte Protokollfelder, die interaktive Filterung und Analyse ermöglichen, ohne Abfragen schreiben zu müssen. CloudWatch Logs erstellt automatisch Facetten für den Namen und den Typ der Datenquelle, und Sie können Facettenrichtlinien für erkannte Felder erstellen, um die Fehlerbehebung zu beschleunigen. Facetten zeigen Werteverteilungen

und Zählungen in CloudWatch Logs Insights an, sodass Muster durch gezielte Erkundung leicht zu erkennen sind.

- **Pipelines:** Erstellen Sie Transformationspipelines, die für alle Logs mit einem bestimmten Datenquellennamen und -typ gelten. Auf diese Weise können Sie konsistente Verarbeitungsregeln für Protokolle aus derselben Quelle definieren.
- **Feldererkennung:** CloudWatch Logs erkennt automatisch Felder und ihre Datentypen für jede Kombination aus Datenquellennamen und -typ auf der Grundlage von Pipeline-Prozessoren. Für AWS verwaltete Protokolle sind Feldstrukturen vordefiniert. Für Anwendungsprotokolle empfehlen wir, konsistente Protokollformate beizubehalten, um die Kompatibilität mit Analysetools wie Amazon S3-Tabellen zu maximieren, die klar definierte Feldstrukturen erfordern.

Mithilfe der `GetLogFields` API können Sie die vollständige Liste der Felder und ihrer Typen für jede Datenquelle einsehen:

```
aws logs get-log-fields --data-source-name <name> --data-source-type <type>
```

Diese Felderkennung und Konsistenz ermöglichen erweiterte Analysen und Integrationen, da externe Tools bei der Verarbeitung Ihrer Protokolldaten mit vorhersehbaren Feldstrukturen arbeiten können.

Unterstützte Datenquellen

CloudWatch Logs unterstützt eine Vielzahl von Datenquellen, darunter AWS Dienste, Sicherheits- und IT-Tools von Drittanbietern sowie benutzerdefinierte Anwendungsquellen. Diese Seite bietet einen umfassenden Katalog aller unterstützten Datenquellen. Informationen darüber, wie Datenquellen erkannt und verwaltet werden, finden Sie unter [the section called “Erkennung und Verwaltung von Datenquellen”](#).

Themen

- [Unterstützt AWS-Services für Datenquellen](#)
- [Unterstützte Quellen von Drittanbietern für Datenquellen](#)
- [Benutzerdefinierte Quellen](#)

Unterstützt AWS-Services für Datenquellen

In der folgenden Tabelle sind diejenigen aufgeführt AWS-Services , die automatisch nach CloudWatch Protokollen als Datenquellen kategorisiert werden:

Name der Datenquelle (@data_source_name - Feld)	Datenquellentyp (@data_source_type -Feld)
amazon_api_gateway	access
amazon_bedrock_agentcore	browser_usage
amazon_bedrock_agentcore	code_interpreter_application
amazon_bedrock_agentcore	code_interpreter_usage
amazon_bedrock_agentcore	gateway_application
amazon_bedrock_agentcore	identity_workload_application
amazon_bedrock_agentcore	memory_application
amazon_bedrock_agentcore	online_evaluation_config
amazon_bedrock_agentcore	runtime_application
amazon_bedrock_agentcore	runtime_usage
amazon_bedrock_agents	application
amazon_bedrock_agents	event
amazon_bedrock_knowledge_bases	application
amazon_cloudfront	access
amazon_cloudfront	connection
amazon_cloudwatch	rum_app_monitor
amazon_cognito	user_pool
amazon_ec2	verified_access
amazon_eks	api_server

Name der Datenquelle (@data_source_name - Feld)	Datenquellentyp (@data_source_type -Feld)
amazon_eks	audit
amazon_eks	authenticator
amazon_eks	controller_manager
amazon_eks	scheduler
amazon_elasticache	cluster
amazon_eventbridge	eventbus_error
amazon_eventbridge	eventbus_info
amazon_eventbridge	pipes_execution
amazon_interactive_video_service	chat
amazon_managed_prometheus	scraper
amazon_managed_prometheus	workspace
amazon_msk	broker
amazon_msk	connect
amazon_opensearch_service	pipeline
amazon_q_business	events
amazon_q_business	sync_job
amazon_q_connect	events
amazon_route53	global_resolver_query
amazon_route53	hosted_zones
amazon_route53	profiles_resolver_query

Name der Datenquelle (@data_source_name - Feld)	Datenquellentyp (@data_source_type -Feld)
amazon_route53	resolver_query
amazon_sagemaker	workteam_activity
amazon_ses	ingress_endpoints
amazon_ses	rule_sets
amazon_ses	traffic_policy
amazon_vpc	flow
amazon_vpc	route_server_peer
amazon_vpc_lattice	access
amazon_vpc_lattice	resource_access
amazon_workmail	access_control
amazon_workmail	authentication
amazon_workmail	personal_access
amazon_workmail	workmail_access
amazon_workmail	workmail_availability
aws_b2b_data_interchange	execution
aws_backup	data_access
aws_backup	hypervisor
aws_clean_rooms	analysis
aws_client_vpn	connection
aws_client_vpn	event

Name der Datenquelle (@data_source_name - Feld)	Datenquellentyp (@data_source_type -Feld)
aws_cloudtrail	data
aws_cloudtrail	management
aws_elemental_mediapackage	egress_access
aws_elemental_mediapackage	ingress_access
aws_elemental_mediatailor	ad_decision
aws_elemental_mediatailor	manifest
aws_elemental_mediatailor	transcode
aws_entity_resolution	id_mapping_workflow
aws_entity_resolution	matching_workflow
aws_iot_fleetwise	error
aws_mainframe_modernization	batch_job
aws_mainframe_modernization	config
aws_mainframe_modernization	console
aws_mainframe_modernization	dataset_import
aws_network_firewall	alert
aws_network_firewall	flow
aws_network_firewall	tls
aws_nlb	access
aws_pcs	job_completion
aws_pcs	scheduler

Name der Datenquelle (@data_source_name - Feld)	Datenquellentyp (@data_source_type -Feld)
aws_security_hub	compliance_finding
aws_security_hub	data_security_finding
aws_security_hub	detection_finding
aws_security_hub	vulnerability_finding
aws_security_hub_cspm	asff_finding
aws_shield	protection_flow
aws_step_functions	express
aws_step_functions	standard
aws_transfer_family	server
aws_waf	access

Unterstützte Quellen von Drittanbietern für Datenquellen

In der folgenden Tabelle sind die Drittanbieterquellen aufgeführt, die automatisch nach CloudWatch Protokollen als Datenquellen kategorisiert werden, wenn sie über Pipelines aufgenommen werden:

Name der Datenquelle (@data_source_name - Feld)	Datenquellentyp (@data_source_type -Feld)
akamai_datastream_2	base_event
akamai_datastream_2	dns_activity
akamai_datastream_2	http_activity
cisco_meraki	api_activity
cisco_meraki	detection_finding

Name der Datenquelle (@data_source_name - Feld)	Datenquellentyp (@data_source_type -Feld)
cisco_meraki	network_activity
cisco_umbrella	data_security_finding
cisco_umbrella	dns_activity
cisco_umbrella	entity_management
cisco_umbrella	network_activity
crowdstrike_falcon	detection_finding
crowdstrike_falcon	process_activity
drupal_core	application_lifecycle
drupal_core	authentication
drupal_core	entity_management
drupal_core	http_activity
entrust_idaas	authentication
entrust_idaas	entity_management
f5_bigip	http_activity
f5_bigip	network_activity
github_auditlogs	account_change
github_auditlogs	api_activity
github_auditlogs	entity_management
microsoft_entraid	account_change
microsoft_entraid	authentication

Name der Datenquelle (@data_source_name - Feld)	Datenquellentyp (@data_source_type -Feld)
microsoft_entraid	entity_management
microsoft_entraid	user_access_management
microsoft_office365	account_change
microsoft_office365	application_lifecycle
microsoft_office365	authentication
microsoft_office365	compliance_finding
microsoft_office365	detection_finding
microsoft_office365	email_activity
microsoft_office365	file_hosting_activity
microsoft_office365	group_management
microsoft_office365	incident_finding
microsoft_office365	user_access_management
microsoft_office365	vulnerability_finding
microsoft_office365	web_resources_activity
microsoft_windows	account_change
microsoft_windows	authentication
microsoft_windows	entity_management
microsoft_windows	event_log_activity
microsoft_windows	file_system_activity
microsoft_windows	group_management

Name der Datenquelle (@data_source_name - Feld)	Datenquellentyp (@data_source_type -Feld)
microsoft_windows	kernel_activity
okta_auth0	api_activity
okta_auth0	authentication
okta_sso	api_activity
okta_sso	authentication
okta_sso	detection_finding
okta_sso	entity_management
onelogin_identity	account_change
onelogin_identity	authentication
onelogin_identity	entity_management
paloaltonetworks_nextgenerationfirewall	authentication
paloaltonetworks_nextgenerationfirewall	detection_finding
paloaltonetworks_nextgenerationfirewall	network_activity
paloaltonetworks_nextgenerationfirewall	process_activity
pingidentity_pingone	account_change
pingidentity_pingone	authentication
pingidentity_pingone	entity_management

Name der Datenquelle (@data_source_name - Feld)	Datenquellentyp (@data_source_type -Feld)
sentinelone_endpointsecurity	dns_activity
sentinelone_endpointsecurity	file_system_activity
sentinelone_endpointsecurity	http_activity
sentinelone_endpointsecurity	process_activity
servicenow_cmdb	api_activity
servicenow_cmdb	datastore_activity
servicenow_cmdb	entity_management
wiz_cnapp	api_activity
wiz_cnapp	authentication
wiz_cnapp	compliance_finding
wiz_cnapp	detection_finding
wiz_cnapp	vulnerability_finding
zeek	authentication
zeek	base_event
zeek	detection_finding
zeek	dhcp_activity
zeek	dns_activity
zeek	email_activity
zeek	ftp_activity
zeek	http_activity

Name der Datenquelle (@data_source_name - Feld)	Datenquellentyp (@data_source_type -Feld)
zeek	network_activity
zeek	rdp_activity
zeek	smb_activity
zeek	software_inventory_info
zeek	ssh_activity
zeek	tunnel_activity
zscaler_internetaccess	authentication
zscaler_internetaccess	dns_activity
zscaler_internetaccess	http_activity
zscaler_internetaccess	network_activity

Zusätzliche Quellen von Drittanbietern über AWS Security Hub CSPM

Zusätzliche Sicherheitserkenntnisse von Drittanbietern sind über die AWS Security Hub CSPM-Integration verfügbar. Die folgenden Partner senden Ergebnisse an Security Hub CSPM, die dann als Datenquellen in CloudWatch Logs verfügbar sind. Umfassende Informationen zu diesen Integrationen finden Sie unter [Third-party Produktintegrationen mit Security Hub CSPM](#) im AWS Security Hub Hub-Benutzerhandbuch.

Partner	Integration
3CoreSec — NTA	Sendet Ergebnisse über Security Hub CSPM
Alert Logic — Bedrohungsmanagement ohne SIEM	Sendet Ergebnisse über Security Hub CSPM

Partner	Integration
Aqua Security — Cloud-native Sicherheitsplattform	Sendet Ergebnisse über Security Hub CSPM
Aqua-Sicherheit — Kube-bench	Sendet Ergebnisse über Security Hub CSPM
Rüstung — Rüstung überall	Sendet Ergebnisse über Security Hub CSPM
AttackIQ	Sendet Ergebnisse über Security Hub CSPM
Barracuda Networks — Hüter der Cloud-Sicherheit	Sendet Ergebnisse über Security Hub CSPM
BigID — BigID Enterprise	Sendet Ergebnisse über Security Hub CSPM
Blaues Sechseck	Sendet Ergebnisse über Security Hub CSPM
Checkpoint — CloudGuard IaaS	Sendet Ergebnisse über Security Hub CSPM
Check Point — Posture Management CloudGuard	Sendet Ergebnisse über Security Hub CSPM
Clarity — Kuppel	Sendet Ergebnisse über Security Hub CSPM
Cloud-Speichersicherheit — Antivirus für Amazon S3	Sendet Ergebnisse über Security Hub CSPM
Contrast Security — Contrast Assess	Sendet Ergebnisse über Security Hub CSPM
CrowdStrike — Falke CrowdStrike	Sendet Ergebnisse über Security Hub CSPM

Partner	Integration
CyberArk — Analyse privilegierter Bedrohungen	Sendet Ergebnisse über Security Hub CSPM
Satz von Daten	Sendet Ergebnisse über Security Hub CSPM
Daten	Sendet Ergebnisse über Security Hub CSPM
Forcepoint — CASB	Sendet Ergebnisse über Security Hub CSPM
Forcepoint — Cloud-Sicherheitsgateway	Sendet Ergebnisse über Security Hub CSPM
Forcepoint — DLP	Sendet Ergebnisse über Security Hub CSPM
Forcepoint — NGFW	Sendet Ergebnisse über Security Hub CSPM
Fuge	Sendet Ergebnisse über Security Hub CSPM
Guardicore — Centra	Sendet Ergebnisse über Security Hub CSPM
HackerOne — Informationen zu Sicherheitslücken	Sendet Ergebnisse über Security Hub CSPM
JFrog — Xray	Sendet Ergebnisse über Security Hub CSPM
Juniper Networks — vSRX-Firewall der nächsten Generation	Sendet Ergebnisse über Security Hub CSPM
k9 Security — Zugriffsanalysator	Sendet Ergebnisse über Security Hub CSPM

Partner	Integration
Schnürsenkelarbeit	Sendet Ergebnisse über Security Hub CSPM
McAfee — MVISION CNAPP	Sendet Ergebnisse über Security Hub CSPM
NETSCOUT — Cyber-Ermittler	Sendet Ergebnisse über Security Hub CSPM
Orca — Cloud-Sicherheitsplattform	Sendet Ergebnisse über Security Hub CSPM
Palo Alto Networks — Prisma Cloud Compute	Sendet Ergebnisse über Security Hub CSPM
Palo Alto Networks — Prisma Cloud Enterprise	Sendet Ergebnisse über Security Hub CSPM
Plerion — Cloud-Sicherheitsplattform	Sendet Ergebnisse über Security Hub CSPM
Prowler	Sendet Ergebnisse über Security Hub CSPM
Qualys — Schwachstellenmanagement	Sendet Ergebnisse über Security Hub CSPM
Rapid7 — InsightVM	Sendet Ergebnisse über Security Hub CSPM
SentinelOne	Sendet Ergebnisse über Security Hub CSPM
Snyk	Sendet Ergebnisse über Security Hub CSPM
Sonrai Security — Sonrai Dig	Sendet Ergebnisse über Security Hub CSPM

Partner	Integration
Sophos — Serverschutz	Sendet Ergebnisse über Security Hub CSPM
StackRox — Kubernetes-Sicherheit	Sendet Ergebnisse über Security Hub CSPM
Sumo Logic — Analyse von Maschinendaten	Sendet Ergebnisse über Security Hub CSPM
Symantec — Schutz von Cloud-Workloads	Sendet Ergebnisse über Security Hub CSPM
Tenable.io	Sendet Ergebnisse über Security Hub CSPM
Trend Micro — Cloud One	Sendet Ergebnisse über Security Hub CSPM
Vectra — Cognito Detect	Sendet Ergebnisse über Security Hub CSPM
Wiz	Sendet Ergebnisse über Security Hub CSPM
Caveonix — Caveonix Cloud	Sendet und empfängt Ergebnisse über Security Hub CSPM
Cloud Custodian	Sendet und empfängt Ergebnisse über Security Hub CSPM
DisruptOps	Sendet und empfängt Ergebnisse über Security Hub CSPM

Partner	Integration
Kion	Sendet und empfängt Ergebnisse über Security Hub CSPM
Turbot	Sendet und empfängt Ergebnisse über Security Hub CSPM

Note

Diese Liste spiegelt die Security Hub-Partnerintegrationen wider, die zum Zeitpunkt der Erstellung dieses Artikels Ergebnisse senden. Da AWS Security Hub regelmäßig neue Partnerintegrationen hinzufügt, finden Sie die aktuelle Liste der verfügbaren Partner unter [Third-party Produktintegrationen mit AWS Security Hub CSPM](#) im Security Hub-Benutzerhandbuch.

Benutzerdefinierte Quellen

Für benutzerdefinierte Anwendungsprotokolle können Sie mithilfe einer der folgenden Methoden Ihre eigene Datenquellen-Kategorisierung definieren:

- Protokollgruppen-Tags — Fügen Sie mithilfe der Schlüssel Tags zu Ihren Protokollgruppen hinzu `cw:datasource:name` und `cw:datasource:type` geben Sie den Namen und den Typ der Datenquelle für alle in der Protokollgruppe aufgenommenen Protokolle an.
- Pipeline-Konfiguration — Konfigurieren Sie Datenquelleninformationen mithilfe von Protokollverarbeitungspipelines, wenn Sie Ihre Anwendungsprotokolle aufnehmen.

Note

Benutzerdefinierte Datenquellennamen dürfen nicht mit „aws“ oder „amazon“ beginnen, um Konflikte mit AWS Serviceprotokollen zu vermeiden.

Logdaten mit CloudWatch Logs Insights analysieren

Mit CloudWatch Logs Insights können Sie Ihre Protokolldaten in Amazon CloudWatch Logs interaktiv suchen und analysieren. Sie können Abfragen durchführen, um effizienter und effektiver auf betriebliche Probleme reagieren zu können. Neben Abfragen mithilfe von Protokollgruppen können Sie Abfragen auch anhand von Facetten, Datenquellen und Datentypen durchführen. Wenn ein Problem auftritt, können Sie CloudWatch Logs Insights verwenden, um mögliche Ursachen zu identifizieren und bereitgestellte Fixes zu überprüfen. Sie sind auf 100 gleichzeitige CloudWatch Logs Insights QL pro Konto beschränkt, einschließlich Abfragen, die zu Dashboards hinzugefügt wurden. Darüber hinaus können Sie 15 Abfragen gleichzeitig für OpenSearch Service PPL oder Service SQL ausführen. OpenSearch

CloudWatch Logs Insights unterstützt drei Abfragesprachen, die Sie für Ihre Abfragen verwenden können:

- Eine speziell entwickelte Logs Insights-Abfragesprache (Logs Insights QL) mit einigen einfachen, aber leistungsstarken Befehlen.
- OpenSearch Service Piped Processing Language (PPL). OpenSearch PPL ermöglicht es Ihnen, Ihre Logs mithilfe einer Reihe von Befehlen zu analysieren, die durch Pipes (|) getrennt sind.

Mit OpenSearch PPL können Sie Daten abrufen, abfragen und analysieren, indem Sie Befehle verwenden, die in der Pipeline zusammengefasst sind, was das Verständnis und die Erstellung komplexer Abfragen erleichtert. Die Syntax ermöglicht die Verkettung von Befehlen zur Transformation und Verarbeitung von Daten. Mit PPL können Sie Daten filtern und aggregieren und eine Vielzahl von mathematischen, Zeichenketten-, Datums-, Bedingungs- und anderen Funktionen für die Analyse verwenden.

- OpenSearch Service Structured Query Language (SQL). Mit OpenSearch SQL-Abfragen können Sie Ihre Logs deklarativ analysieren. Sie können Befehle wie SELECT, FROM, WHERE, GROUP BY, HAVING und verschiedene andere Befehle und Funktionen verwenden, die in SQL verfügbar sind. Sie können JOINS über Protokollgruppen hinweg ausführen, Daten mithilfe von Unterabfragen protokollübergreifend korrelieren und die umfangreiche Palette von JSON-, mathematischen, String-, Conditional- und anderen SQL-Funktionen verwenden, um leistungsstarke Analysen von Protokollen durchzuführen.

Wenn Sie entweder SQL- oder PPL-Befehle verwenden, achten Sie darauf, Felder mit Sonderzeichen (nicht alphabetisch und nicht numerisch) in Backticks einzuschließen, um sie erfolgreich abzufragen. Schließen Sie beispielsweise, und in Backticks ein. @message

`Operation.ExportTest::Field` Sie müssen Felder mit rein alphabetischen Namen nicht in Backticks einschließen.

CloudWatch Logs Insights bietet die folgenden Funktionen, die für die Verwendung mit allen Abfragesprachen verfügbar sind.

- Automatische [Erkennung von Protokollfeldern](#) in Protokollen von AWS Diensten wie Amazon Route 53, AWS Lambda AWS CloudTrail, und Amazon VPC sowie in allen Anwendungen oder benutzerdefinierten Protokollen, die Protokollereignisse als JSON ausgeben.
- Erstellung von [Feldindizes](#), um Kosten zu senken und Ergebnisse zu beschleunigen, insbesondere bei Abfragen einer großen Anzahl von Protokollgruppen oder Protokollereignissen. Nachdem Sie Feldindizes von Feldern erstellt haben, die in Ihren Protokollereignissen häufig vorkommen, können Sie sie in einer Abfrage verwenden. Die Abfrage überspringt die Verarbeitung von Protokollereignissen, von denen bekannt ist, dass sie das indizierte Feld nicht enthalten, und verarbeitet weniger Daten.

 Note

Der `filterIndex` Befehl ist nur in Logs Insights QL verfügbar.

- [Erkennung und Analyse von Mustern](#) in Ihren Protokollereignissen. Ein Muster ist eine gemeinsame Textstruktur, die sich in Ihren Protokollfeldern wiederholt. Wenn Sie sich die Ergebnisse einer Abfrage ansehen, können Sie die Registerkarte Muster wählen, um sich die Muster anzusehen, die CloudWatch Logs anhand einer Stichprobe Ihrer Ergebnisse gefunden hat.
- [Abfragen speichern](#), Ihren Abfrageverlauf einsehen, gespeicherte Abfragen erneut ausführen und [gespeicherte Abfragen mit Parametern verwenden](#).
- [Hinzufügen von Abfragen zu Dashboards](#).
- [Verschlüsseln von Abfrageergebnissen](#) mit AWS Key Management Service
- [Durch die Abfragegenerierung in natürlicher Sprache](#) können Sie CloudWatch Logs Insights-Abfragen in natürlicher Sprache erstellen. Sie können Fragen zu den gesuchten Daten stellen oder diese beschreiben. Anschließend generiert die KI auf der Grundlage Ihrer Aufforderung eine Abfrage und erklärt Zeile für Zeile, wie die Abfrage funktioniert.
- [Verwenden Sie Facetten, um Ihre Logs zu gruppieren, zu filtern und interaktiv](#) zu erkunden.
- Mithilfe von [Umgebungsprotokollen](#) können Sie Protokollzeilen vor und nach einem bestimmten Protokolldatensatz in Ihren Abfrageergebnissen anzeigen, um sofort einen Überblick über kritische Protokollereignisse zu erhalten. Sie können den Bereich so konfigurieren, dass 5, 10, 20, 50 oder

100 Protokollzeilen vor und nach einem ausgewählten Datensatz angezeigt werden, und in den umliegenden Protokollen nach bestimmten Schlüsselwörtern suchen.

Die folgenden CloudWatch Logs Insights-Funktionen werden nur unterstützt, wenn Sie Logs Insights QL verwenden.

- [Vergleichsabfragen](#), die Protokollereignisse in einer Protokollgruppe mit Protokollereignissen aus einem früheren Zeitraum vergleichen.

 Important

CloudWatch Logs Insights kann nicht auf Protokollereignisse zugreifen, deren Zeitstempel vor der Erstellung der Protokollgruppe liegen.

Wenn Sie mit einem Konto angemeldet sind, das als Überwachungskonto für CloudWatch kontoübergreifende Observability eingerichtet wurde, können Sie CloudWatch Logs Insights-Abfragen für Protokollgruppen in Quellkonten ausführen, die mit diesem Überwachungskonto verknüpft sind. Sie können eine Abfrage ausführen, die mehrere Protokollgruppen abfragt, die sich in verschiedenen Konten befinden. Weitere Informationen finden Sie unter [Kontoübergreifende Beobachtbarkeit von CloudWatch](#).

Wenn Sie Abfragen mit Logs Insights QL erstellen, können Sie CloudWatch Logs Insights-Abfragen auch in natürlicher Sprache erstellen. Stellen Sie dazu Fragen zu den Daten, nach denen Sie suchen, oder beschreiben Sie sie. Diese AI-assisted Funktion generiert auf der Grundlage Ihrer Eingabeaufforderung eine Abfrage und erklärt Zeile für Zeile, wie die Abfrage funktioniert. Weitere Informationen finden Sie unter [Verwenden natürlicher Sprache zum Generieren und Aktualisieren von CloudWatch Logs Insights-Abfragen](#).

Abfragen, die eine der unterstützten Abfragesprachen verwenden, laufen nach 60 Minuten ab, wenn sie nicht abgeschlossen wurden. Abfrageergebnisse sind sieben Tage lang verfügbar.

CloudWatch Für Logs Insights-Abfragen fallen Gebühren an, die auf der abgefragten Datenmenge basieren, unabhängig von der Abfragesprache. Weitere Informationen finden Sie unter [CloudWatch Amazon-Preise](#).

Sie können CloudWatch Logs Insights verwenden, um nach Protokolldaten zu suchen, die am 5. November 2018 oder später an CloudWatch Logs gesendet wurden.

Important

Wenn Ihr Netzwerksicherheitsteam die Verwendung von Web-Sockets nicht zulässt, können Sie derzeit nicht auf den CloudWatch Logs Insights-Bereich der CloudWatch Konsole zugreifen. Sie können die CloudWatch Logs Insights-Abfragefunktionen mithilfe von APIs verwenden. Weitere Informationen finden Sie [StartQuery](#) in der Amazon CloudWatch Logs API-Referenz.

Inhalt

- [Unterstützte Abfragesprachen](#)
- [Verwenden Sie natürliche Sprache, um CloudWatch Logs Insights-Abfragen zu generieren und zu aktualisieren](#)
- [Unterstützte Protokolle und erkannte Felder](#)
- [Erstellen Sie Feldindizes, um die Abfrageleistung zu verbessern und das Scanvolumen zu reduzieren](#)
- [Verwenden Sie Facetten, um Logs zu gruppieren und zu untersuchen](#)
- [Umliegende Logs in CloudWatch Logs Insights anzeigen](#)
- [Musteranalyse](#)
- [Speichern Sie CloudWatch Logs Insights-Abfragen und führen Sie sie erneut aus](#)
- [Abfrage zum Dashboard hinzufügen oder Abfrageergebnisse exportieren](#)
- [Anzeigen von laufenden Abfragen oder Abfrageverlauf](#)
- [Verschlüsseln Sie Abfrageergebnisse mit AWS Key Management Service](#)
- [Generieren Sie eine Zusammenfassung in natürlicher Sprache aus den CloudWatch Logs Insights-Abfrageergebnissen](#)

Unterstützte Abfragesprachen

In den folgenden Abschnitten sind die Befehle aufgeführt, die in den einzelnen Abfragesprachen unterstützt werden. Sie beschreiben auch das Syntaxformat und enthalten Beispielabfragen.

Themen

- [CloudWatch Logs Insights-Abfragesprache \(Logs Insights QL\)](#)
- [OpenSearch Piped Processing Language \(PPL\)](#)

- [OpenSearch Strukturierte Abfragesprache \(SQL\)](#)

CloudWatch Logs Insights-Abfragesprache (Logs Insights QL)

Dieser Abschnitt enthält eine vollständige Dokumentation der Befehle und Funktionen von Logs Insights QL. Er enthält auch Beispielabfragen für diese Sprache.

Informationen zu anderen Abfragesprachen, die Sie verwenden können, finden Sie unter [OpenSearch Service PPL](#), [OpenSearch Service SQL](#) und [CloudWatch Metrics Insights](#).

Themen

- [CloudWatch Abfragesyntax in Logs Insights](#)
- [Erste Schritte mit Logs Insights QL: Query tutorials](#)
- [Beispielabfragen](#)
- [Vergleiche \(Diff\) mit früheren Zeitbereichen](#)
- [Visualisieren von Protokolldaten in Diagrammen](#)

CloudWatch Abfragesyntax in Logs Insights

Dieser Abschnitt enthält Einzelheiten zur Logs Insights QL. Die Abfragesyntax unterstützt verschiedene Funktionen und Operationen, die unter anderem allgemeine Funktionen, Arithmetik- und Vergleichsoperationen sowie reguläre Ausdrücke beinhalten.

Important

Beachten Sie die folgenden bewährten Methoden, um zu vermeiden, dass durch die Ausführung umfangreicher Abfragen übermäßige Gebühren anfallen:

- Wählen Sie für jede Abfrage nur die erforderlichen Protokollgruppen aus.
- Geben Sie immer den engstmöglichen Zeitraum für Ihre Abfragen an.
- Wenn Sie die Konsole zum Ausführen von Abfragen verwenden, brechen Sie alle Abfragen ab, bevor Sie die CloudWatch Logs Insights-Konsolenseite schließen. Andernfalls werden Abfragen so lange ausgeführt, bis sie abgeschlossen sind.
- Wenn Sie einem Dashboard ein CloudWatch Logs Insights-Widget hinzufügen, stellen Sie sicher, dass das Dashboard nicht mit hoher Frequenz aktualisiert wird, da bei jeder Aktualisierung eine neue Abfrage gestartet wird.

Zum Erstellen von Abfragen, die mehrere Befehle enthalten, trennen Sie die Befehle durch einen senkrechten Strich (|).

Zum Erstellen von Abfragen, die Kommentare enthalten, kennzeichnen Sie die Kommentare durch ein Rautenzeichen (#).

 Note

CloudWatch Logs Insights erkennt automatisch Felder für verschiedene Protokolltypen und generiert Felder, die mit dem @-Zeichen beginnen. Weitere Informationen zu diesen Feldern finden Sie unter [Unterstützte Protokolle und entdeckte Felder](#) im CloudWatchAmazon-Benutzerhandbuch.

In der folgenden Tabelle wird jeder Befehl kurz beschrieben. Im Anschluss an diese Tabelle finden Sie eine ausführlichere Beschreibung der einzelnen Befehle mit Beispielen.

 Note

Alle QL-Abfragebefehle von Logs Insights werden für Protokollgruppen der Standard-Protokollklasse unterstützt. Protokollgruppen der Protokollklasse für seltenen Zugriff unterstützen alle Logs Insights QL-Abfragebefehle mit Ausnahme von `patterndiff`, `undunmask`.

<u>anomaly</u>	Identifiziert mithilfe von maschinellem Lernen ungewöhnliche Muster in Ihren Protokolldaten.
<u>display</u>	Zeigt ein bestimmtes Feld oder bestimmte Felder in den Abfrageergebnissen an.
<u>fields</u>	Zeigt bestimmte Felder in Abfrageergebnissen an und unterstützt Funktionen und Vorgänge mit denen Sie Feldwerte ändern und neue Felder zur Verwendung in Ihrer Abfrage erstellen können.
<u>filter</u>	Filtert die Abfrage so, dass nur die Protokollereignisse zurückgegeben werden, die mindestens einer Bedingung entsprechen.

filterIndex	Erzwingt eine Abfrage, nur die Protokollgruppen zu scannen, die sowohl für das in einem Feldindex erwähnte Feld indexiert sind als auch einen Wert für diesen Feldindex enthalten. Dadurch wird das gescannte Volumen reduziert, indem versucht wird, nur Protokollereignisse aus diesen Protokollgruppen zu scannen, die den in der Abfrage für diesen Feldindex angegebenen Wert enthalten. Dieser Befehl wird für Protokollgruppen der Protokollklasse für seltenen Zugriff nicht unterstützt.
pattern	Gruppieren Sie Ihre Protokolldaten automatisch in Muster. Ein Muster ist eine gemeinsame Textstruktur, die sich in Ihren Protokollfeldern wiederholt. CloudWatch Logs Insights bietet Ihnen Möglichkeiten, die in Ihren Protokollereignissen gefundenen Muster zu analysieren. Weitere Informationen finden Sie unter Musteranalyse.
diff	Vergleichen Sie die in Ihrem angeforderten Zeitraum gefundenen Protokollereignisse mit den Protokollereignissen aus einem früheren Zeitraum gleicher Länge, sodass Sie nach Trends suchen und herausfinden können, ob bestimmte Protokollereignisse neu sind.
parse	Extrahieren Sie Daten aus einem Protokollfeld, um ein extrahiertes Feld zu erstellen, das Sie in Ihrer Abfrage verarbeiten können. parse unterstützt sowohl den globalen Modus mit Platzhaltern als auch reguläre Ausdrücke.
sort	Zeigen Sie die zurückgegebenen Protokollereignisse in aufsteigender (asc) oder absteigender (desc) Reihenfolge an.
SOURCE	Die Aufnahme SOURCE in eine Abfrage ist eine nützliche Methode, um eine große Anzahl von Protokollgruppen anzugeben, die in eine Abfrage aufgenommen werden sollen, und zwar auf der Grundlage des Protokollgruppennamens, der Kontokennungen, der Protokollgruppenklasse, der Datenquellen oder der Protokollgruppen-Tags. Dieser Befehl wird nur unterstützt, wenn Sie eine Abfrage in der AWS CLI oder programmgesteuert erstellen, nicht in der Konsole. CloudWatch

<u>stats</u>	Berechnet aggregierte Statistiken anhand der Werte in den Protokollfeldern.
<u>limit</u>	Gibt die maximale Anzahl der Protokollereignisse an, die Ihre Abfrage zurückgeben soll. Nützlich in Verbindung mit sort , um „Top 20“ oder „letzte 20 Ergebnisse“ zu erhalten.
<u>dedup</u>	Entfernt doppelte Ergebnisse auf der Grundlage bestimmter Werte in von Ihnen angegebenen Feldern.
<u>unmask</u>	Zeigt den gesamten Inhalt eines Protokollereignisses an, bei dem einige Inhalte aufgrund einer Datenschutzrichtlinie maskiert wurden. Weitere Informationen zum Datenschutz in Protokollgruppen finden Sie unter Den Schutz vertraulicher Protokolldaten mit Maskierung unterstützen .
<u>unnest</u>	Reduziert eine als Eingabe verwendete Liste, sodass mehrere Datensätze mit einem einzigen Datensatz für jedes Element in der Liste erstellt werden.
<u>lookup</u>	Reichert Protokollereignisse mit Daten aus einer Nachschlagetabelle an, indem Feldwerte abgeglichen werden. Verwenden Sie Nachschlagetabellen, um Referenzdaten wie Benutzerdetails, Anwendungsnamen oder Produktinformationen zu Ihren Abfrageergebnissen hinzuzufügen.
<u>join</u>	Kombiniert Protokollereignisse aus einer Quellprotokollgruppe mit Ereignissen aus einer anderen Protokollgruppe oder einem Abfrageergebnis, das auf einem passenden Feld basiert. Verwenden Sie den Befehl <code>join</code> , um verwandte Protokollereignisse aus verschiedenen Quellen zu korrelieren, indem Sie Schlüssel verwenden, die allen gemeinsam sind, z. B. übereinstimmende Anforderungskennungen oder Transaktions-IDs.
<u>subqueries</u>	Eine Unterabfrage ist eine verschachtelte Logs Insights-Abfrage, die als Eingabe für eine andere Abfrage verwendet werden kann. Unterabfragen können verwendet werden, um Zwischenergebnismengen abzuleiten, die dann von nachfolgenden Befehlen verwendet werden.

[Weitere Vorgänge und Funktionen](#)

CloudWatch Logs Insights unterstützt auch viele Funktionen und Operationen für Vergleiche, Arithmetik, Datetime, Zahlen, Zeichenketten, IP-Adressen und allgemeine Funktionen und Operationen.

In den folgenden Abschnitten finden Sie weitere Informationen zu den CloudWatch Logs Insights-Abfragebefehlen.

Themen

- [Logs Insights QL-Befehle werden in Protokollklassen unterstützt](#)
- [Anomalie](#)
- [display](#)
- [fields](#)
- [Filter](#)
- [filterIndex](#)
- [SOURCE](#)
- [pattern](#)
- [diff](#)
- [parse](#)
- [relevante Felder](#)
- [expand](#)
- [sort](#)
- [stats](#)
- [limit](#)
- [dedup](#)
- [unmask](#)
- [unnest](#)
- [Nachschlagen](#)
- [join](#)
- [Unterabfragen](#)
- [Boolesche, Vergleichs-, numerische, Datetime- und andere Funktionen](#)
- [Felder, die Sonderzeichen enthalten](#)

- [Alias und Kommentare in Abfragen verwenden](#)

Logs Insights QL-Befehle werden in Protokollklassen unterstützt

Alle Logs Insights QL-Abfragebefehle werden für Protokollgruppen in der Standard-Protokollklasse unterstützt. Protokollgruppen der Protokollklasse für seltenen Zugriff unterstützen alle Abfragebefehle außer `pattern`, `difffilterIndex`, und `unmask`.

Anomalie

Wird verwendet `anomaly`, um mithilfe von maschinellem Lernen automatisch ungewöhnliche Muster und potenzielle Probleme in Ihren Protokolldaten zu identifizieren.

Der `anomaly` Befehl erweitert die bestehende `pattern` Funktionalität und nutzt fortschrittliche Analysen, um potenzielle Anomalien in Protokolldaten zu identifizieren. Sie können `anomaly` damit den Zeitaufwand für die Identifizierung und Lösung betrieblicher Probleme reduzieren, indem Sie automatisch ungewöhnliche Muster oder Verhaltensweisen in Ihren Protokollen aufdecken.

Der `anomaly` Befehl arbeitet mit dem [pattern](#) Befehl zusammen, um zunächst Protokollmuster zu identifizieren und dann Anomalien innerhalb dieser Muster zu erkennen. Sie können es auch `anomaly` mit den [sort](#) Befehlen [filter](#) oder kombinieren, um die Anomalieerkennung auf bestimmte Teilmengen Ihrer Daten zu konzentrieren.

Befehlseingabe für Anomalien

Der `anomaly` Befehl wird normalerweise nach dem [pattern](#) Befehl verwendet, um die in Ihren Protokolldaten identifizierten Muster zu analysieren. Der Befehl benötigt keine zusätzlichen Parameter und analysiert die Ausgabe der vorherigen Befehle in Ihrer Abfrage.

Arten der identifizierten Anomalien

Der `anomaly` Befehl identifiziert fünf verschiedene Arten von Anomalien:

- Anomalien bei der Musterfrequenz: Ungewöhnliche Häufigkeiten bestimmter Protokollmuster, z. B. wenn eine Anwendung anfängt, mehr Fehlermeldungen als üblich zu generieren.
- Neue Musteranomalien: Bisher unbekannte Protokollmuster, die auf neue Arten von Fehlern oder Meldungen in Ihren Protokollen hinweisen können.
- Anomalien bei Token-Variationen: Unerwartete Änderungen im Inhalt von Protokollnachrichten, die auf ungewöhnliche Variationen der erwarteten Protokollformate hinweisen können.

- Numerische Token-Anomalien: Ungewöhnliche Änderungen der numerischen Werte in Protokollen, die dazu beitragen können, potenzielle Leistungsprobleme oder unerwartete metrische Abweichungen zu erkennen.
- HTTP-Fehlercode-Anomalien: Muster im Zusammenhang mit HTTP-Fehlerantworten, besonders nützlich bei der Überwachung von Webanwendungen und APIs.

Ausgabe des Befehls „Anomalie“

Der `anomaly` Befehl behält alle Felder der Eingabedaten bei und fügt Ergebnisse zur Erkennung von Anomalien hinzu, um ungewöhnliche Muster in Ihren Protokolldaten zu identifizieren.

Beispiele

Der folgende Befehl identifiziert Muster in Ihren Protokolldaten und erkennt dann Anomalien innerhalb dieser Muster:

```
fields @timestamp, @message
| pattern @message
| anomaly
```

Der `anomaly` Befehl kann zusammen mit der Filterung verwendet werden, um sich auf bestimmte Protokolltypen zu konzentrieren:

```
fields @timestamp, @message
| filter @type = "REPORT"
| pattern @message
| anomaly
```

Der `anomaly` Befehl kann mit der Sortierung kombiniert werden, um die Ergebnisse zu organisieren:

```
fields @timestamp, @message
| filter @type = "ERROR"
| pattern @message
| anomaly
| sort @timestamp desc
```

display

Verwenden Sie `display`, um ein bestimmtes Feld oder bestimmte Felder in Abfrageergebnissen anzuzeigen.

Der `display`-Befehl zeigt nur die von Ihnen angegebenen Felder an. Wenn Ihre Abfrage mehrere `display`-Befehle enthält, zeigen die Abfrageergebnisse nur das Feld/die Felder, das/die Sie im letzten `display`-Befehl angegeben haben.

Beispiel: Ein Feld anzeigen

Das Code-Snippet zeigt ein Beispiel für eine Abfrage, die den `parse`-Befehl verwendet, um Daten aus `@message` zu extrahieren und so die extrahierten Felder `loggingType` und `loggingMessage` zu erstellen. Die Abfrage gibt Protokollereignisse zurück, bei denen die Werte für `loggingType` FEHLER sind. `display` zeigt nur die Werte für `loggingMessage` in den Abfrageergebnissen an.

```
fields @message
| parse @message "[*] *" as loggingType, loggingMessage
| filter loggingType = "ERROR"
| display loggingMessage
```

Tip

Verwenden Sie `display` nur einmal in einer Abfrage. Wenn Sie `display` mehr als einmal in Ihrer Abfrage verwenden, zeigen die Abfrageergebnisse nur das Feld an, das Sie bei der letzten Nutzung des `display`-Befehls angegeben haben.

fields

Verwenden Sie `fields`, um bestimmte Felder in Abfrageergebnissen anzuzeigen.

Wenn Ihre Abfrage mehrere `fields`-Befehle und keinen `display`-Befehl enthält, werden in den Ergebnissen alle Felder angezeigt, die in den `fields`-Befehlen angegeben sind.

Beispiel: Bestimmte Felder anzeigen

Das folgende Beispiel zeigt eine Abfrage, die 20 Protokollereignisse zurückgibt und sie in absteigender Reihenfolge anordnet. Die Werte für `@timestamp` und `@message` werden in den Abfrageergebnissen angezeigt.

```
fields @timestamp, @message
| sort @timestamp desc
| limit 20
```

Verwenden Sie `fields` anstelle `vondisplay`, wenn Sie die verschiedenen Funktionen und Operationen verwenden möchten, die von `fields` zum Ändern von Feldwerten und zum Erstellen neuer Felder, die in Abfragen verwendet werden können, unterstützt werden.

Sie können den `fields`-Befehl mit dem Schlüsselwort `as` verwenden, um extrahierte Felder zu erstellen, die Felder und Funktionen in Ihren Protokollereignissen verwenden. Beispielsweise erstellt `fields ispresent as isRes` ein extrahiertes Feld mit dem Namen `isRes` und das extrahierte Feld kann für den Rest Ihrer Abfrage verwendet werden.

Filter

Verwenden Sie `filter`, um Protokollereignisse abzurufen, die einer oder mehreren Bedingungen entsprechen.

Beispiel: Filtern von Protokollereignissen mithilfe einer Bedingung

Das Code-Snippet zeigt ein Beispiel für eine Abfrage, die alle Protokollereignisse zurückgibt, deren Wert für `range` größer als 3 000 ist. Die Abfrage begrenzt die Ergebnisse auf 20 Protokollereignisse und sortiert die Protokollereignisse nach `@timestamp` und in absteigender Reihenfolge.

```
fields @timestamp, @message
| filter (range>3000)
| sort @timestamp desc
| limit 20
```

Beispiel: Filtern von Protokollereignissen mit mehr als einer Bedingung

Sie können die Schlüsselwörter `and` und `or` verwenden, um mehr als eine Bedingung zu kombinieren.

Der Codeausschnitt zeigt ein Beispiel für Protokollereignisse, bei denen der Wert für `range` größer als 3 000 und der Wert für `accountId` gleich 123 456 789 012 ist. Die Abfrage begrenzt die Ergebnisse auf 20 Protokollereignisse und sortiert die Protokollereignisse nach `@timestamp` und in absteigender Reihenfolge.

```
fields @timestamp, @message
| filter (range>3000 and accountId=123456789012)
| sort @timestamp desc
| limit 20
```

Indizierte Felder und der Filterbefehl

Wenn Sie Feldindizes für eine Protokollgruppe erstellt haben, können Sie diese Feldindizes nutzen, um Ihre `filter` Abfragen effizienter zu gestalten und das gescannte Volumen zu reduzieren. Nehmen wir beispielsweise an, Sie haben einen Feldindex für `requestId` erstellt. Dann jede CloudWatch Logs Insights-Abfrage für diese Protokollgruppe, die Protokollereignisse enthält `filter requestId = value` oder versucht, `filter requestId IN [value, value, ...]` die Verarbeitung von Protokollereignissen zu überspringen, von denen bekannt ist, dass sie das indizierte Feld nicht enthalten. Wenn Sie versuchen, nur die Protokollereignisse zu scannen, von denen bekannt ist, dass sie dieses indizierte Feld enthalten, kann das Scanvolumen reduziert und die Abfrage beschleunigt werden.

Weitere Informationen zu Feldindizes und deren Erstellung finden Sie unter. [Erstellen Sie Feldindizes, um die Abfrageleistung zu verbessern und das Scanvolumen zu reduzieren](#)

Important

Nur Abfragen mit `filter fieldName = ...` und `filter fieldName IN ...` profitieren von den Verbesserungen des Feldindexes. Abfragen mit `filter fieldName like` verwenden keine Indizes und scannen immer alle Protokollereignisse in den ausgewählten Protokollgruppen.

Beispiel: Suchen Sie mithilfe von Indizes nach Protokollereignissen, die sich auf eine bestimmte Anforderungs-ID beziehen

In diesem Beispiel wird davon ausgegangen, dass Sie einen Feldindex für `requestId` erstellt haben. Bei Protokollgruppen, die diesen Feldindex verwenden, versucht die Abfrage anhand von Feldindizes, die geringste Anzahl von Protokollereignissen nach Ereignissen `requestId` mit einem Wert von `123456` zu durchsuchen.

```
fields @timestamp, @message
| filter requestId = "123456"
| limit 20
```

Übereinstimmungen und Reguläre Ausdrücke im Filterbefehl

Der Filterbefehl unterstützt die Verwendung regulärer Ausdrücke. Sie können die folgenden Vergleichsoperatoren (=, !=, <, <=, >, >=) und Booleschen Operatoren (and, or und not) verwenden.

Sie können das Schlüsselwort `in` verwenden, um auf eine festgelegte Mitgliedschaft zu testen und auf Elemente in einem Array zu prüfen. Platzieren Sie das Array nach `in`, um auf Elemente in einem Array zu prüfen. Sie können den Booleschen Operator `not` mit `in` verwenden. Sie können Abfragen erstellen, die `in` verwenden, um Protokollereignisse zurückzugeben, die mit Zeichenfolgen übereinstimmen. Die Felder müssen vollständige Zeichenfolgen sein. Der folgende Codeausschnitt zeigt beispielsweise eine Abfrage, die `in` verwendet, um Protokollereignisse zurückzugeben, wobei das Feld `logGroup` die vollständige Zeichenfolge `example_group` ist.

```
fields @timestamp, @message
| filter logGroup in ["example_group"]
```

Sie können die Schlüsselwortausdrücke `like` und `not like` für den Abgleich mit Teilzeichenfolgen verwenden. Sie können den Operator für reguläre Ausdrücke `=~` für den Abgleich mit Teilzeichenfolgen verwenden. Schließen Sie für den Abgleich einer Teilzeichenfolge mit `like` und `not like` die Teilzeichenfolge, die abgeglichen werden soll, in einfache oder doppelte Anführungszeichen ein. Sie können Muster von regulären Ausdrücken mit `like` und `not like` verwenden. Um eine Teilzeichenfolge mit dem Operator für reguläre Ausdrücke abzugleichen, schließen Sie die Teilzeichenfolge, die Sie abgleichen möchten, in Schrägstriche ein. Die folgenden Beispiele enthalten Code-Snippets, die zeigen, wie Sie Teilzeichenfolgen mit dem Befehl `filter` abgleichen können.

Beispiele: Abgleich von Teilzeichenfolgen

Die folgenden drei Beispiele geben Protokollereignisse zurück, bei denen `f1` das Wort `Exception` (Ausnahme) enthält. Bei allen drei Beispielen muss die Groß-/Kleinschreibung beachtet werden.

Das erste Beispiel gleicht eine Teilzeichenfolge mit `like` ab.

```
fields f1, f2, f3
| filter f1 like "Exception"
```

Das zweite Beispiel gleicht eine Teilzeichenfolge mit `like` und einem Muster von regulären Ausdrücken ab.

```
fields f1, f2, f3
| filter f1 like /Exception/
```

Das dritte Beispiel gleicht eine Teilzeichenfolge mit einem regulären Ausdruck ab.

```
fields f1, f2, f3
| filter f1 =~ /Exception/
```

Beispiel: Abgleichen von Teilzeichenfolgen mit Platzhaltern

Sie können das Punktsymbol (.) als Platzhalter in regulären Ausdrücken verwenden, um Teilzeichenfolgen abzugleichen. Im folgenden Beispiel gibt die Abfrage Übereinstimmungen zurück, bei denen der Wert für f1 mit der Zeichenfolge ServiceLog beginnt.

```
fields f1, f2, f3
| filter f1 like /ServiceLog./
```

Sie können ein Sternchen nach dem Punktsymbol (.*) platzieren, um einen gierigen Quantifizierer zu erstellen, der so viele Übereinstimmungen wie möglich zurückgibt. Beispielsweise gibt die folgende Abfrage Übereinstimmungen zurück, bei denen der Wert für f1 nicht nur mit der Zeichenfolge ServiceLog beginnt, sondern auch die Zeichenfolge ServiceLog beinhaltet.

```
fields f1, f2, f3
| filter f1 like /ServiceLog.*/
```

Mögliche Übereinstimmungen können wie folgt formatiert werden:

- ServiceLogSampleApiLogGroup
- SampleApiLogGroupServiceLog

Beispiel: Ausschließen von Teilzeichenfolgen aus Abgleichen

Das folgende Beispiel zeigt eine Abfrage, die Protokollereignisse zurückgibt, bei denen f1 nicht das Wort Exception (Ausnahme) enthält. Das Beispiel berücksichtigt Groß- und Kleinschreibung.

```
fields f1, f2, f3
| filter f1 not like "Exception"
```

Beispiel: Abgleichen von Teilzeichenfolgen mit Mustern, die Groß- und Kleinschreibung nicht berücksichtigen

Sie können Teilzeichenfolgen, bei denen die Groß-/Kleinschreibung nicht beachtet wird, mit `like` und regulären Ausdrücken abgleichen. Platzieren Sie den folgenden Parameter `(?i)` vor der Teilzeichenfolge, die Sie abgleichen möchten. Das folgende Beispiel zeigt eine Abfrage, die Protokollereignisse zurückgibt, bei denen `f1` das Wort `Exception` (Ausnahme) oder `exception` (ausnahme) enthält.

```
fields f1, f2, f3
| filter f1 like /(?!i)Exception/
```

filterIndex

Wird verwendet, `filterIndex` um nur indizierte Daten zurückzugeben, indem eine Abfrage erzwungen wird, nur Protokollgruppen zu scannen, die für ein Feld indiziert sind, das Sie in der Abfrage angeben. Für diese Protokollgruppen, die in diesem Feld indexiert sind, wird die Abfrage weiter optimiert, indem die Protokollgruppen übersprungen werden, die keine Protokollereignisse enthalten, die das in der Abfrage für das indizierte Feld angegebene Feld enthalten. Das gescannte Volumen wird weiter reduziert, indem versucht wird, nur Protokollereignisse aus diesen Protokollgruppen zu scannen, die dem in der Abfrage für diesen Feldindex angegebenen Wert entsprechen. Weitere Informationen zu Feldindizes und deren Erstellung finden Sie unter [Erstellen Sie Feldindizes, um die Abfrageleistung zu verbessern und das Scanvolumen zu reduzieren](#).

`filterIndex` mithilfe indizierter Felder können Sie Protokollgruppen, die Petabyte an Protokolldaten enthalten, effizient abfragen, indem der tatsächliche Suchbereich auf Protokollgruppen und Protokollereignisse mit Feldindizes beschränkt wird.

Nehmen wir beispielsweise an, Sie haben für einige Protokollgruppen `IPAddress` in Ihrem Konto einen Feldindex erstellt. Sie können dann die folgende Abfrage erstellen und alle Protokollgruppen im Konto abfragen, um nach Protokollereignissen zu suchen, die den Wert `198.51.100.0` im `IPAddress` Feld enthalten.

```
fields @timestamp, @message
| filterIndex IPAddress = "198.51.100.0"
| limit 20
```

Der `filterIndex` Befehl veranlasst diese Abfrage, alle Protokollgruppen zu überspringen, für `IPAddress` die kein Index erstellt wurde. Darüber hinaus überspringt die Abfrage innerhalb der

indizierten Protokollgruppen Protokollereignisse, die zwar ein IPaddress Feld haben, aber nicht 198.51.100.0 als Wert für dieses Feld beobachtet wurden.

Verwenden Sie den IN Operator, um die Ergebnisse auf einen beliebigen Wert für die indizierten Felder zu erweitern. Im folgenden Beispiel werden Protokollereignisse gefunden, die entweder den Wert 198.51.100.0 oder das 198.51.100.1 IPaddress Feld enthalten.

```
fields @timestamp, @message
| filterIndex IPaddress in ["198.51.100.0", "198.51.100.1"]
| limit 20
```

CloudWatch Logs stellt Standardfeldindizes für alle Protokollgruppen in der Standard-Protokollklasse bereit. Standardfeldindizes sind automatisch für die folgenden Felder verfügbar:

- @logStream
- @aws.region
- @aws.account
- @source.log
- @data_source_name
- @data_source_type
- @data_format
- traceId
- severityText
- attributes.session.id

CloudWatch Logs bietet auch Standardfeldindizes für bestimmte Kombinationen von Datenquellennamen und -typen. Standardfeldindizes sind automatisch für die folgenden Kombinationen von Datenquellennamen und -typ verfügbar:

Name und Typ der Datenquelle	Standard-Feldindizes
amazon_vpc.flow	action
	logStatus
	region

Name und Typ der Datenquelle	Standard-Feldindizes
	flowDirection type
amazon_route53.resolver_query	query_type transport rcode
aws_waf.access	action httpRequest.country
aws_cloudtrail.data	eventSource
aws_cloudtrail.management	eventName awsRegion userAgent errorCode eventType managementEvent readOnly eventCategory requestId

Standardfeldindizes gelten zusätzlich zu allen benutzerdefinierten Feldindizes, die Sie in Ihrer Richtlinie definieren. Standardfeldindizes werden nicht auf Ihr [Feldindexkontingent](#) angerechnet.

FilterIndex im Vergleich zu Filter

Sehen Sie sich die folgenden Beispielabfragen an `filter`, um den Unterschied zwischen `filterIndex` und zu verdeutlichen. Gehen Sie davon aus `IPAddress`, dass Sie einen Feldindex für vier Ihrer Protokollgruppen erstellt haben, aber nicht für eine fünfte Protokollgruppe. Bei der folgenden Abfrage `filterIndex` wird das Scannen der Protokollgruppe übersprungen, für die das Feld nicht indexiert ist. Für jede indizierte Protokollgruppe wird versucht, nur Protokollereignisse zu scannen, bei denen das indizierte Feld vorhanden ist, und es werden auch nur Ergebnisse zurückgegeben, die nach der Erstellung des Feldindexes erstellt wurden.

```
fields @timestamp, @message
| filterIndex IPAddress = "198.51.100.0"
| limit 20
```

Wenn Sie dagegen `filter` statt `filterIndex` für eine Abfrage derselben fünf Protokollgruppen verwenden, versucht die Abfrage, nicht nur die Protokollereignisse zu scannen, die den Wert in den indizierten Protokollgruppen enthalten, sondern auch die fünfte Protokollgruppe, die nicht indexiert ist, und scannt jedes Protokollereignis in dieser fünften Protokollgruppe.

```
fields @timestamp, @message
| filter IPAddress = "198.51.100.0"
| limit 20
```

SOURCE

Die Aufnahme `SOURCE` in eine Abfrage ist eine nützliche Methode, um die and/or Datenquellen der Protokollgruppen anzugeben, die in eine Abfrage aufgenommen werden sollen, wenn Sie die AWS CLI OR-API zum Erstellen einer Abfrage verwenden. Der `SOURCE` Befehl wird nur in der AWS CLI AND-API unterstützt, nicht in der CloudWatch Konsole. Wenn Sie die CloudWatch Konsole verwenden, um eine Abfrage zu starten, verwenden Sie die Konsolenschnittstelle, um die Protokollgruppen anzugeben.

Loggruppen abfragen

Um die abzufragenden Protokollgruppen anzugeben, können Sie die folgenden Schlüsselwörter verwenden: `SOURCE`

- `namePrefix` führt die Abfrage für Protokollgruppen aus, deren Namen mit der von Ihnen angegebenen Zeichenfolge beginnen. Wenn Sie dies weglassen, werden alle Protokollgruppen abgefragt.

Sie können bis zu fünf Präfixe in die Liste aufnehmen.

- `accountIdentifier` führt die Abfrage für Protokollgruppen im angegebenen AWS Konto aus. Dies funktioniert nur, wenn Sie die Abfrage in einem Überwachungskonto ausführen. Wenn Sie dies weglassen, werden standardmäßig alle verknüpften Quellkonten und das aktuelle Überwachungskonto abgefragt. [Weitere Informationen zur kontenübergreifenden Beobachtbarkeit finden Sie unter kontenübergreifende CloudWatch Beobachtbarkeit.](#)

Sie können bis zu 20 Konto-Identifikatoren in die Liste aufnehmen.

- `logGroupClass` führt die Abfrage anhand von Protokollgruppen aus, die sich in der angegebenen Protokollklasse (Standard oder Infrequent Access) befinden. Wenn Sie dies weglassen, wird die Standardprotokollklasse Standard verwendet. Weitere Hinweise zu Protokollklassen finden Sie unter [Klassen protokollieren](#).

Da Sie eine große Anzahl von Protokollgruppen angeben können, die auf diese Weise abgefragt werden sollen, empfehlen wir, diese `SOURCE` nur in Abfragen zu verwenden, die von Ihnen erstellte Feldindizes nutzen. Weitere Informationen zur Indizierung von Feldern in Protokollgruppen finden Sie unter [Erstellen Sie Feldindizes, um die Abfrageleistung zu verbessern und das Scanvolumen zu reduzieren](#)

Im folgenden Beispiel werden alle Protokollgruppen im Konto ausgewählt. Wenn es sich um ein Überwachungskonto handelt, werden die Protokollgruppen für alle Überwachungs- und alle Quellkonten ausgewählt. Wenn die Gesamtzahl der Protokollgruppen 10.000 überschreitet, wird eine Fehlermeldung angezeigt, in der Sie aufgefordert werden, die Anzahl der Protokollgruppen mithilfe einer anderen Methode zur Auswahl der Protokollgruppen zu reduzieren.

```
SOURCE logGroups()
```

Im folgenden Beispiel werden die Protokollgruppen im 111122223333 Quellkonto ausgewählt. Wenn Sie eine Abfrage in einem Überwachungskonto mit CloudWatch kontenübergreifender Observability starten, werden standardmäßig Protokollgruppen in allen Quellkonten und im Überwachungskonto ausgewählt.

```
SOURCE logGroups(accountIdentifier:['111122223333'])
```

Im nächsten Beispiel werden Protokollgruppen anhand von Namenspräfixen ausgewählt.

```
SOURCE logGroups(namePrefix: ['namePrefix1', 'namePrefix2'])
```

Im folgenden Beispiel werden alle Protokollgruppen in der Protokollklasse für seltenen Zugriff ausgewählt. Wenn Sie den `class` Bezeichner nicht angeben, gilt die Abfrage nur für Protokollgruppen in der Standard-Protokollklasse, was die Standardprotokollklasse ist.

```
SOURCE logGroups(class: ['INFREQUENT_ACCESS'])
```

Im nächsten Beispiel werden Protokollgruppen im Konto 111122223333 ausgewählt, die mit bestimmten Namenspräfixen beginnen und zur Standard-Protokollklasse gehören. Die Klasse wird im Befehl nicht erwähnt, da Standard der Standardwert für die Protokollklasse ist.

```
SOURCE logGroups(accountIdentifier:['111122223333'], namePrefix: ['namePrefix1', 'namePrefix2'])
```

Das letzte Beispiel zeigt, wie der `SOURCE` Befehl zusammen mit dem `start-query` AWS CLI Befehl verwendet wird.

```
aws logs start-query
--region us-east-1
--start-time 1729728200
--end-time 1729728215
--query-string "SOURCE logGroups(namePrefix: ['Query']) | fields @message | limit 5"
```

Datenquellen abfragen

Um die abzufragenden Datenquellen anzugeben, können Sie das `dataSource` Schlüsselwort verwenden. `SOURCE` Sie können bis zu zehn Datenquellen in die Liste aufnehmen.

Im folgenden Beispiel wird die `amazon_vpc.flow` Datenquelle ausgewählt.

```
SOURCE dataSource(['amazon_vpc.flow'])
```

Im folgenden Beispiel wird die `amazon_vpc.flow` Datenquelle ausgewählt und die Anzahl der Protokollgruppen auf der Grundlage eines Protokollgruppennamenpräfixes begrenzt.

```
SOURCE dataSource(['amazon_vpc.flow']) logGroups(namePrefix: ['namePrefix1'])
```

Fragen Sie Protokollgruppen nach Tags ab

Verwenden Sie die `logGroupTags` Funktion, `SOURCE` um Protokollgruppen nach ihren Tags zu filtern. Geben Sie Tags als eine Liste von Tagfiltern mit jeweils einem `key` optionalen `values` Array an.

- Mehrere Tagfilter mit unterschiedlichen Schlüsseln werden mit der UND-Logik kombiniert.
- Mehrere Werte innerhalb desselben Tagfilters werden mit der ODER-Logik kombiniert.
- Wird `*` für den Platzhalterabgleich verwendet. Entspricht beispielsweise `payment*` Werten, die mit `payment` beginnen.
- Verwenden Sie `!` es als Präfix für Negation. Entspricht beispielsweise Werten `!production`, die dies nicht `production` sind.
- Sie können bis zu fünf Tagfilter mit jeweils bis zu fünf Werten verwenden.

Im folgenden Beispiel werden alle Protokollgruppen ausgewählt, die mit gekennzeichnet sind `team=team1` OR `team=team2`.

```
SOURCE logGroupTags([{"key":"team", "values":["team1", "team2"]}] )
| fields @message, @timestamp
```

Im nächsten Beispiel werden Protokollgruppen ausgewählt `payment`, bei denen das `service` Tag mit `beginnt` und das `environment` Tag nicht `production`.

```
SOURCE logGroupTags([{"key":"service", "values":["payment*"]}, {"key":"environment",
"values":["!production"]}])
| fields @message, @timestamp
```

Im folgenden Beispiel wird die Tagfilterung mit einem Namenspräfixfilter kombiniert.

```
SOURCE logGroups(namePrefix: ['/aws/lambda']) logGroupTags([{"key":"environment",
"values":["production"]}])
| fields @message, @timestamp
```

pattern

Verwenden Sie `pattern`, um Ihre Protokolldaten automatisch nach Mustern zu clustern.

Ein Muster ist eine gemeinsame Textstruktur, die sich in Ihren Protokollfeldern wiederholt. Sie können `pattern` es verwenden, um neue Trends aufzudecken, bekannte Fehler zu überwachen und häufig auftretende oder kostspielige Protokollzeilen zu identifizieren. CloudWatch Logs Insights bietet

auch eine Konsolenoberfläche, mit der Sie Muster in Ihren Protokollereignissen finden und weiter analysieren können. Weitere Informationen finden Sie unter [Musteranalyse](#).

Da der `pattern` Befehl häufig auftretende Muster automatisch identifiziert, können Sie ihn als Ausgangspunkt für die Suche und Analyse Ihrer Logs verwenden. Sie können auch `pattern` mit den Befehlen [filter](#), [parse](#) oder [sort](#) kombinieren, um Muster in detaillierteren Abfragen zu identifizieren.

Eingabe des Muster-Befehls

Der `pattern`-Befehl erwartet eine der folgenden Eingaben: das `@message`-Feld, ein mit dem [parse](#)-Befehl erstelltes extrahiertes Feld oder eine Zeichenfolge, die mit einer oder mehreren [Zeichenfolgenfunktionen](#) manipuliert wurde.

Wenn CloudWatch Logs nicht ableiten kann, für welche Art von Daten ein dynamisches Token steht, zeigt es es als `<Token- number > number` an und gibt an, an welcher Stelle im Muster dieses Token im Vergleich zu den anderen dynamischen Token vorkommt.

Zu den häufigsten Beispielen für dynamische Token gehören Fehlercodes, IP-Adressen, Zeitstempel und Anforderungs-IDs.

Ausgabe des Muster-Befehls

Die Ausgabe des Befehls `pattern` sieht wie folgt aus:

- `@pattern`: Eine gemeinsam genutzte Textstruktur, die sich in Ihren Protokollfeldern wiederholt. Felder, die innerhalb eines Musters variieren, z. B. eine Anforderungs-ID oder ein Zeitstempel, werden durch Token dargestellt. Wenn CloudWatch Logs ermitteln kann, für welchen Datentyp ein dynamisches Token steht, wird das Token als `<string-number>` angezeigt. Dies *string* ist eine Beschreibung des Datentyps, für den das Token steht. Das *number* zeigt, an welcher Stelle des Musters dieses Token im Vergleich zu den anderen dynamischen Tokens erscheint.

CloudWatch Logs weist die Zeichenfolge als Teil des Namens auf der Grundlage der Analyse des Inhalts der Protokollereignisse zu, in denen der Name enthalten ist.

Wenn CloudWatch Logs nicht ableiten kann, für welche Art von Daten ein dynamisches Token steht, zeigt es es als `<Token- number > number` an und gibt an, an welcher Stelle im Muster dieses Token im Vergleich zu den anderen dynamischen Token vorkommt.

Zum Beispiel ist `[INFO] Request time: <Time-1> ms` eine mögliche Ausgabe für die Protokollmeldung `[INFO] Request time: 327 ms`.

- `@ratio`: Das Verhältnis von Protokollereignissen aus einem ausgewählten Zeitraum und angegebenen Protokollgruppen, die einem identifizierten Muster entsprechen. Wenn beispielsweise die Hälfte der Protokollereignisse in den ausgewählten Protokollgruppen und im ausgewählten Zeitraum mit dem Muster übereinstimmt, gibt `@ratio 0.50` aus
- `@sampleCount`: Eine Zählung der Anzahl der Protokollereignisse aus einem ausgewählten Zeitraum und bestimmten Protokollgruppen, die einem bestimmten Muster entsprechen.
- `@severityLabel`: Der Schweregrad oder die Stufe des Protokolls, der die Art der in einem Protokoll enthaltenen Informationen angibt. Beispiele: `Error`, `Warning`, `Info` oder `Debug`.

Beispiele

Der folgende Befehl identifiziert Protokolle mit ähnlichen Strukturen in der/den angegebenen Protokollgruppe(n) über den ausgewählten Zeitraum und gruppiert sie nach Muster und Anzahl

```
pattern @message
```

Der `pattern`-Befehl kann in Kombination mit dem [filter](#)-Befehl verwendet werden

```
filter @message like /ERROR/  
| pattern @message
```

Der `pattern`-Befehl kann mit den Befehlen [parse](#) und [sort](#) verwendet werden

```
filter @message like /ERROR/  
| parse @message 'Failed to do: *' as cause  
| pattern cause  
| sort @sampleCount asc
```

diff

Vergleicht die in Ihrem angeforderten Zeitraum gefundenen Protokollereignisse mit den Protokollereignissen aus einem früheren Zeitraum gleicher Länge. Auf diese Weise können Sie nach Trends suchen und herausfinden, ob bestimmte Protokollereignisse neu sind.

Fügen Sie dem `diff` Befehl einen Modifikator hinzu, um den Zeitraum anzugeben, mit dem Sie vergleichen möchten:

- `diff` vergleicht die Protokollereignisse im aktuell ausgewählten Zeitraum mit den Protokollereignissen des unmittelbar vorhergehenden Zeitraums.

- `diff previousDay` vergleicht die Protokollereignisse im aktuell ausgewählten Zeitraum mit den Protokollereignissen aus derselben Zeit des Vortages.
- `diff previousWeek` vergleicht die Protokollereignisse im aktuell ausgewählten Zeitraum mit den Protokollereignissen aus derselben Zeit der Vorwoche.
- `diff previousMonth` vergleicht die Protokollereignisse im aktuell ausgewählten Zeitraum mit den Protokollereignissen aus derselben Zeit des Vormonats.

Weitere Informationen finden Sie unter [Vergleiche \(Diff\) mit früheren Zeitbereichen](#).

parse

Wird verwendet, um Daten aus einem Protokollfeld zu extrahieren und extrahierte Felder zu erstellen, die Sie in Ihrer Abfrage verarbeiten können. Der `parse` Befehl unterstützt drei Modi: Glob-Ausdrücke, reguläre Ausdrücke und `logfmt`.

Wenn weggelassen `fieldName` wird, `@message` wird es standardmäßig verwendet. Sie können aus jedem benannten Feld eine Analyse durchführen, indem Sie den Feldnamen als erstes Argument angeben.

Wenn ein Protokollereignis nicht dem angegebenen Muster entspricht, wird es immer noch in den Ergebnissen angezeigt, jedoch ohne die extrahierten Felder.

Glob-Modus

Verwenden Sie Platzhalter (*) als Platzhalter für Werte, die Sie extrahieren möchten, und weisen Sie sie benannten Feldern zu. `as`

Syntax

```
parse fieldName "pattern" as alias1, alias2
```

Die Anzahl der * Platzhalter muss der Anzahl der Aliase entsprechen.

Beispiele

```
parse @message "user=*, method:*, latency := *" as @user,  
@method, @latency | stats avg(@latency) by @method, @user
```

```
parse @logStream "*/**/*" as env, service, instance, shard
```

```
| stats count(*) by env, service
```

Verkettete Analyse

Extrahieren Sie ein Feld und analysieren Sie das extrahierte Feld anschließend weiter.

```
parse @message "url=*" as url
| parse url "/api/*/users/*" as apiVersion, userId
| display apiVersion, userId
```

Regex-Modus

Verwenden Sie einen regulären Ausdruck mit benannten Capture-Gruppen, um Felder zu extrahieren. Hinweise zur Syntax regulärer Ausdrücke finden Sie unter [Unterstützte Syntax für reguläre Ausdrücke \(Regex\)](#).

Syntax

```
parse fieldName /regex/
```

Verwenden Sie benannte Capture-Gruppen(`?<name>...`), um extrahierte Felder zu definieren.

Beispiele

Verwenden Sie benannte Erfassungsgruppen, um Felder zu extrahieren

```
parse @message /user=(?<user2>.*?), method:(?<method2>.*?),
  latency := (?<latency2>.*?)/ | stats avg(latency2) by @method2,
  @user2
```

Verwenden Sie eine benannte Capture-Gruppe, um die ENI aus einem VPC-Flow-Protokoll zu extrahieren

```
parse @message /(?<NetworkInterface>eni-.*?) /
| display NetworkInterface, @message
```

Logfmt-Modus

Wird verwendet `parse logfmt`, um Logzeilen im Logfmt-Format in Schlüssel-Wert-Paare zu zerlegen. Logfmt ist ein strukturiertes Logging-Format, bei dem jede Zeile durch Leerzeichen getrennte Paare enthält. `key=value`

Syntax

```
parse fieldName logfmt as alias
```

Das Ergebnis ist eine Map, auf die Sie mit Punktnotation zugreifen (z. B.,`lf.level`). `lf.msg`

Beispiele

```
parse @message logfmt as lf
| filter lf.level = "error"
| display lf.msg, lf.duration
```

```
parse @message logfmt as lf
| stats count(*) by lf.host
```

relevante Felder

Wird verwendet `relevantfields`, um automatisch zu ermitteln, welche Felder in Ihren Protokolldaten für eine bestimmte Bedingung am relevantesten sind. Der Befehl vergleicht die Basisprotokolle mit der Teilmenge, die Ihrer Bedingung entspricht, und gibt Felder zurück, die nach ihrer Relevanzbewertung geordnet sind.

Syntax

```
relevantfields [field1, field2, ...] where condition
```

Die Feldliste ist optional. Wenn sie weggelassen wird, werden alle Felder analysiert.

Der Befehl gibt die folgenden Ausgabefelder zurück:

- `@fieldName`— Name des Feldes
- `@relevanceScore`— Ein Wert, der die Relevanz auf einer Skala von 0 bis 1 angibt
- `@topRelevanceContributors`— Zeigt für kategoriale Felder die Einträge mit den höchsten Häufigkeitsverschiebungen an.
- `@conditionalMedian`— Bei numerischen Feldern der Median der Werte in Logs, die die Bedingung erfüllen
- `@baselineMedian`— Bei numerischen Feldern der Median der Werte in Logs, die die Bedingung nicht erfüllen

Beispiele

Analysieren Sie alle Felder auf langsame API-Antworten

```
relevantfields where Time > 400
```

Konzentrieren Sie die Analyse auf bestimmte Felder

```
relevantfields Controller, Path, Service where Time > 400
```

Identifizieren Sie die Treiber von Lambda-Timeouts

```
relevantfields where Duration > 5000
```

expand

Wird verwendet `expand`, um aus einem Feld, das ein JSON-Array enthält, ein separates Protokollereignis für jedes Element im Array zu erstellen. Alle anderen Felder aus dem ursprünglichen Protokollereignis werden in jedem neuen Ereignis dupliziert.

Syntax

```
expand fieldName
```

Beispiel

Wenn ein Protokollereignis `items = ["apple","banana","cherry"]` und `enthält host = "web-01"`, werden drei Protokollereignisse `expand items` erzeugt: `{items: "apple", host: "web-01"}`, `{items: "banana", host: "web-01"}`, und `{items: "cherry", host: "web-01"}`.

```
expand items  
| stats count(*) by items, host
```

sort

Verwenden Sie `sort`, um Protokollereignisse in aufsteigender (`asc`) oder absteigender (`desc`) Reihenfolge nach einem bestimmten Feld anzuzeigen. Sie können dies zusammen mit dem Befehl `limit` verwenden, um „Obere N“- oder „Untere N“-Abfragen zu erstellen.

Der Sortieralgorithmus ist eine aktualisierte Version der natürlichen Sortierung. Wenn Sie in aufsteigender Reihenfolge sortieren, wird die folgende Logik verwendet.

- Alle Werte, die keine Zahlen sind, stehen vor allen Zahlenwerten. Zahlenwerte sind Werte, die nur Zahlen enthalten, keine Mischung aus Zahlen und anderen Zeichen.
- Bei Werten, die keine Zahlen sind, gruppiert der Algorithmus aufeinanderfolgende numerische Zeichen und aufeinanderfolgende alphabetische Zeichen zum Vergleich in separate Blöcke. Er ordnet nichtnumerische Teile nach ihren Unicode-Werten und numerische Teile zuerst nach ihrer Länge und dann nach ihrem numerischen Wert.

Weitere Informationen zur Unicode-Reihenfolge finden Sie unter [Liste der Unicode-Zeichen](#).

Das Folgende ist beispielsweise das Ergebnis einer Sortierung in aufsteigender Reihenfolge.

[illegible]

Wenn Sie in absteigender Reihenfolge sortieren, sind die Sortierergebnisse umgekehrt.

Die folgende Abfrage für Flussprotokolle von Amazon VPC findet zum Beispiel die ersten 15 Paketübertragungen zwischen Hosts.

```
stats sum(packets) as packetsTransferred by srcAddr, dstAddr
| sort packetsTransferred desc
```

```
| limit 15
```

stats

Verwenden Sie `stats`, um Visualisierungen Ihrer Protokolldaten zu erstellen, z. B. Balkendiagramme, Liniendiagramme und gestapelte Flächendiagramme. Auf diese Weise können Sie Muster in Ihren Protokolldaten effizienter identifizieren. CloudWatch Logs Insights generiert Visualisierungen für Abfragen, die die `stats` Funktion und eine oder mehrere Aggregationsfunktionen verwenden.

Beispielsweise gibt die folgende Abfrage in einer Route-53-Protokollgruppe Visualisierungen zurück, die die Verteilung der Route-53-Datensätze pro Stunde nach Abfragetyp zeigen.

```
stats count(*) by queryType, bin(1h)
```

Mit all diesen Abfragen können Balkendiagramme erzeugt werden. Wenn Ihre Abfrage die Funktion `bin()` verwendet, um Daten nach einem einzelnen Feld im Zeitverlauf zu gruppieren, können Sie auch Liniendiagramme und gestapelte Flächendiagramme anzeigen.

Für die Funktion `bin` werden folgende Zeiteinheiten und Abkürzungen unterstützt. Alle Einheiten und Abkürzungen, die mehr als ein Zeichen enthalten, können durch Hinzufügen von „s“ pluralisiert werden. Somit kann sowohl `hr` als auch `hrs` verwendet werden, um Stunden anzugeben.

- `millisecond ms msec`
- `second s sec`
- `minute m min`
- `hour h hr`
- `day d`
- `week w`
- `month mo mon`
- `quarter q qtr`
- `year y yr`

Themen

- [Visualisieren von Zeitreihendaten](#)
- [Visualisieren von nach Feldern gruppierten Protokolldaten](#)

- [Verwenden mehrerer Statistikbefehle in einer einzigen Abfrage](#)
- [Funktionen zur Verwendung mit „stats“](#)

Visualisieren von Zeitreihendaten

Visualisierungen von Zeitreihen funktionieren für Abfragen mit folgenden Merkmalen:

- Die Abfrage enthält eine oder mehrere Aggregationsfunktionen. Weitere Informationen finden Sie unter [Aggregation Functions in the Stats Command](#).
- Die Abfrage verwendet die `bin()`-Funktion. Damit können Sie die Daten nach einem Feld gruppieren.

Diese Abfragen können Linien-, Flächen-, Balken- und Kreisdiagramme erzeugen.

Beispiele

Ein vollständiges Tutorial finden Sie unter [the section called “Tutorial: Ausführen einer Abfrage, die eine Visualisierung von Zeitreihen erzeugt”](#).

Im Folgenden finden Sie weitere Beispielabfragen, die für die Zeitreihenvisualisierung funktionieren.

Die folgende Abfrage erzeugt eine Visualisierung der Durchschnittswerte des `myfield1`-Felds an, mit einem Datenpunkt, der alle fünf Minuten erstellt wird. Jeder Datenpunkt ist die Aggregation der Durchschnitte der `myfield1`-Werte aus den Protokollen der letzten fünf Minuten.

```
stats avg(myfield1) by bin(5m)
```

Die folgende Abfrage erzeugt eine Visualisierung von drei Werten basierend auf verschiedenen Feldern, wobei alle fünf Minuten ein Datenpunkt erstellt wird. Die Visualisierung wird erzeugt, weil die Abfrage Aggregationsfunktionen enthält und `bin()` als Gruppierungsfeld verwendet.

```
stats avg(myfield1), min(myfield2), max(myfield3) by bin(5m)
```

Einschränkungen von Linien- und gestapelten Flächendiagrammen

Abfragen, die Protokolleintragsinformationen aggregieren, die Funktion `bin()` jedoch nicht verwenden, können Balkendiagramme generieren. Die Abfragen können jedoch keine Liniendiagramme oder gestapelte Flächendiagramme generieren. Weitere Informationen zu diesen

Abfragetypen finden Sie unter [the section called “Visualisieren von nach Feldern gruppierten Protokolldaten”](#).

Visualisieren von nach Feldern gruppierten Protokolldaten

Sie können Balkendiagramme für Abfragen erstellen, die die `stats`-Funktion und eine oder mehrere Aggregationsfunktionen verwenden. Weitere Informationen finden Sie unter [Aggregation Functions in the Stats Command](#).

Führen Sie die Abfrage aus, um die Visualisierung aufzurufen. Wählen Sie dann die Registerkarte Visualisierung aus, klicken Sie auf den Pfeil neben Linie und auf Balken. Visualisierungen sind mit maximal 100 Balken im Balkendiagramm beschränkt.

Beispiele

Ein vollständiges Tutorial finden Sie unter [the section called “Tutorial: Ausführen einer Abfrage, die eine nach Protokollfeldern gruppierte Visualisierung erzeugt”](#). Die folgenden Absätze enthalten weitere Beispielabfragen für die Visualisierung nach Feldern.

Die folgende VPC-Flow-Protokollabfrage ermittelt die durchschnittliche Anzahl von Bytes, die pro Sitzung für die einzelnen Zieladressen übertragen werden.

```
stats avg(bytes) by dstAddr
```

Sie können auch ein Diagramm erstellen, das mehr als einen Balken für jeden resultierenden Wert enthält. Die folgende VPC-Flow-Protokollabfrage ermittelt beispielsweise die durchschnittliche und maximale Anzahl von Bytes, die pro Sitzung an die einzelnen Zieladressen übertragen werden.

```
stats avg(bytes), max(bytes) by dstAddr
```

Die folgende Abfrage ermittelt die Anzahl der Amazon-Route-53-Abfrageprotokolle für jeden Abfragetyp.

```
stats count(*) by queryType
```

Verwenden mehrerer Statistikbefehle in einer einzigen Abfrage

Sie können mehrere `stats` Befehle in einer einzigen Abfrage verwenden. Auf diese Weise können Sie zusätzliche Aggregationen für die Ausgabe der ersten Aggregation durchführen. Die maximale

Anzahl von `stats` Befehlen, die in einer Abfrage zulässig sind, hängt von der Protokollklasse der Protokollgruppe ab.

Beispiel: Abfrage mit zwei **stats**-Befehlen

Die folgende Abfrage ermittelt beispielsweise zunächst das gesamte Verkehrsaufkommen in 5-Minuten-Abschnitten und berechnet dann das höchste, niedrigste und durchschnittliche Verkehrsaufkommen unter diesen 5-Minuten-Abschnitten.

```
FIELDS strlen(@message) AS message_length
| STATS sum(message_length)/1024/1024 as logs_mb BY bin(5m)
| STATS max(logs_mb) AS peak_ingest_mb,
      min(logs_mb) AS min_ingest_mb,
      avg(logs_mb) AS avg_ingest_mb
```

Beispiel: Kombinieren Sie mehrere Statistikbefehle mit anderen Funktionen wie **filter**, **fields**, **bin**

Sie können mehrere `stats` Befehle mit anderen Befehlen wie `filter` und `fields` in einer einzigen Abfrage kombinieren. Die folgende Abfrage ermittelt beispielsweise die Anzahl der unterschiedlichen IP-Adressen in Sitzungen und ermittelt die Anzahl der Sitzungen nach Clientplattform, filtert diese IP-Adressen und ermittelt schließlich den Durchschnitt der Sitzungsanfragen pro Clientplattform.

```
STATS count_distinct(client_ip) AS session_ips,
      count(*) AS requests BY session_id, client_platform
| FILTER session_ips > 1
| STATS count(*) AS multiple_ip_sessions,
      sum(requests) / count(*) AS avg_session_requests BY client_platform
```

Sie können die `bin`- und `dateceil`-Funktionen in Abfragen mit mehreren `stats`-Befehlen verwenden. Die folgende Abfrage fasst beispielsweise Nachrichten zunächst zu 5-Minuten-Blöcken zusammen, aggregiert diese 5-Minuten-Blöcke dann zu 10-Minuten-Blöcken und berechnet das höchste, niedrigste und durchschnittliche Verkehrsaufkommen innerhalb jedes 10-Minuten-Blocks.

```
FIELDS strlen(@message) AS message_length
| STATS sum(message_length) / 1024 / 1024 AS logs_mb BY BIN(5m) as @t
| STATS max(logs_mb) AS peak_ingest_mb,
      min(logs_mb) AS min_ingest_mb,
      avg(logs_mb) AS avg_ingest_mb BY dateceil(@t, 10m)
```

Hinweise und Einschränkungen

Die maximale Anzahl von stats Befehlen in einer Abfrage hängt von der Protokollklasse ab:

- Standard-Protokollklasse: maximal 10 stats Befehle pro Abfrage
- Protokollklasse für seltenen Zugriff: maximal 2 stats Befehle pro Abfrage

Diese Kontingente können nicht geändert werden.

Wenn Sie einen limit Befehl sort oder verwenden, muss er nach dem letzten stats Befehl erscheinen. Wenn sie vor dem letzten stats Befehl steht, ist die Abfrage nicht gültig.

Wenn eine Abfrage mehrere stats Befehle enthält, werden die Teilergebnisse der Abfrage erst angezeigt, wenn die erste stats Aggregation abgeschlossen ist.

In nachfolgenden stats Befehlen in einer einzelnen Abfrage können Sie nur auf Felder verweisen, die im vorherigen stats Befehl definiert wurden. Die folgende Abfrage ist beispielsweise nicht gültig, da das @message-Feld nach der ersten stats-Aggregation nicht verfügbar sein wird.

```
FIELDS @message
| STATS SUM(Fault) by Operation
# You can only reference `SUM(Fault)` or Operation at this point
| STATS MAX(strlen(@message)) AS MaxMessageSize # Invalid reference to @message
```

Alle Felder, auf die Sie nach einem stats Befehl verweisen, müssen in diesem stats Befehl definiert sein.

```
STATS sum(x) as sum_x by y, z
| STATS max(sum_x) as max_x by z
# You can only reference `max(sum_x)`, max_x or z at this point
```

Important

Die bin-Funktion verwendet das @timestamp-Feld immer implizit. Das bedeutet, dass Sie es nicht bin in einem nachfolgenden stats Befehl verwenden können, ohne den vorherigen stats Befehl zur Weitergabe des timestamp Felds zu verwenden. Beispielsweise ist die folgende Abfrage nicht zulässig.

```
FIELDS strlen(@message) AS message_length
| STATS sum(message_length) AS ingested_bytes BY @logStream
| STATS avg(ingested_bytes) BY bin(5m) # Invalid reference to @timestamp field
```

Definieren Sie das `@timestamp` Feld stattdessen im vorherigen `stats` Befehl, und Sie können es dann zusammen mit `dateceil` einem nachfolgenden `stats` Befehl verwenden, wie im folgenden Beispiel.

```
FIELDS strlen(@message) AS message_length
| STATS sum(message_length) AS ingested_bytes, max(@timestamp) as @t BY
@logStream
| STATS avg(ingested_bytes) BY dateceil(@t, 5m)
```

Funktionen zur Verwendung mit „stats“

CloudWatch Logs Insights unterstützt sowohl Funktionen zur Aggregation von Statistiken als auch Funktionen ohne Aggregation von Statistiken.

Sie können `statsaggregation`-Funktionen im Befehl `stats` sowie als Argumente für andere Funktionen verwenden.

Funktion	Ergebnistyp	Description
<code>avg(fieldName: NumericLogField)</code>	number	Der Mittelwert der Werte im angegebenen Feld.
<code>count()</code> <code>count(fieldName: LogField)</code>	number	Zählt die Protokollereignisse. <code>count()</code> (oder <code>count(*)</code>) zählt alle von der Abfrage zurückgegebenen Ereignisse, während <code>count(fieldName)</code> alle Datensätze, die den angegebenen Feldnamen enthalten, zählt.
<code>count_distinct(fieldName: LogField)</code>	number	Liefert die Anzahl der eindeutigen Werte für das Feld. Wenn das Feld eine sehr hohe Kardinalität hat (zahlreiche eindeutige Werte enthält), ist der von <code>count_distinct</code> zurückgegebene Wert lediglich eine Annäherung.
<code>max(fieldName: LogField)</code>	LogFieldValue	Das Maximum der Werte für dieses Protokollfeld in den abgefragten Protokollen.

Funktion	Ergebnistyp	Description
<code>min(fieldName: LogField)</code>	LogFieldValue	Das Minimum der Werte für dieses Protokollfeld in den abgefragten Protokollen.
<code>pct(fieldName: LogFieldValue, percent: number)</code>	LogFieldValue	Ein Perzentil gibt die relative Stelle eines Wertes in einer Datenmenge an. Zum Beispiel gibt <code>pct(@duration, 95)</code> den <code>@duration</code> -Wert zurück, bei dem 95 Prozent der Werte von <code>@duration</code> niedriger als dieser Wert und 5 Prozent höher als dieser Wert sind.
<code>stddev(fieldName: NumericLogField)</code>	number	Die Standardabweichung der Werte im angegebenen Feld.
<code>sum(fieldName: NumericLogField)</code>	number	Die Summe der Werte im angegebenen Feld.

Statistik-Nicht-Aggregations-Funktionen

Sie können Nicht-Aggregationsfunktionen im Befehl `stats` sowie als Argumente für andere Funktionen verwenden.

Funktion	Ergebnistyp	Description
<code>earliest(fieldName: LogField)</code>	LogField	Gibt den Wert von <code>fieldName</code> aus dem Protokollereignis mit dem frühesten Zeitstempel in den abgefragten Protokollen zurück.
<code>latest(fieldName: LogField)</code>	LogField	Gibt den Wert von <code>fieldName</code> aus dem Protokollereignis mit dem neuesten Zeitstempel in den abgefragten Protokollen zurück.
<code>sortsFirst(fieldName: LogField)</code>	LogField	Gibt den Wert von <code>fieldName</code> zurück, der in der Sortierung der abgefragten Protokolle an erster Stelle steht.

Funktion	Ergebnistyp	Description
<code>sortsLast(fieldName: LogField)</code>	LogField	Gibt den Wert von <code>fieldName</code> zurück, der in der Sortierung der abgefragten Protokolle an letzter Stelle steht.

limit

Verwenden Sie `limit`, um die Anzahl der Protokollereignisse anzugeben, die Ihre Abfrage zurückgeben soll. Wenn Sie diese Option weglassen, gibt die Abfrage standardmäßig bis zu 10.000 Protokollereignisse zurück. Sie können einen `limit` Wert von bis zu 100.000 angeben.

Das folgende Beispiel gibt beispielsweise nur die letzten 25 Protokollereignisse zurück:

```
fields @timestamp, @message | sort @timestamp desc | limit 25
```

dedup

Verwenden Sie `dedup`, um doppelte Ergebnisse auf der Grundlage bestimmter Werte in von Ihnen angegebenen Feldern zu entfernen. Sie können `dedup` mit einem oder mehreren Feldern verwenden. Wenn Sie ein Feld mit `dedup` angeben, wird für jeden eindeutigen Wert dieses Felds nur ein Protokollereignis zurückgegeben. Wenn Sie mehrere Felder angeben, wird für jede eindeutige Kombination von Werten für diese Felder ein Protokollereignis zurückgegeben.

Duplikate werden auf der Grundlage der Sortierreihenfolge verworfen, wobei nur das erste Ergebnis in der Sortierreihenfolge beibehalten wird. Wir empfehlen, Ihre Ergebnisse zu sortieren, bevor Sie den Befehl `dedup` dafür ausführen. Wenn die Ergebnisse nicht sortiert werden, bevor sie den Befehl `dedup` durchlaufen, wird die absteigende Standard-Sortierreihenfolge mit `@timestamp` verwendet.

Nullwerte werden bei der Auswertung nicht als Duplikate betrachtet. Protokollereignisse mit Nullwerten für eines der angegebenen Felder werden beibehalten. Um Felder mit Nullwerten zu eliminieren, verwenden Sie **filter** mit der Funktion `isPresent(field)`.

Der einzige Abfragebefehl, den Sie in einer Abfrage nach dem Befehl `dedup` verwenden können, ist `limit`.

Wenn Sie die Option `dedup` in einer Abfrage verwenden, zeigt die Konsole eine Meldung wie „X von Y“-Datensätzen anzeigen an, wobei X für die Anzahl der deduplizierten Ergebnisse und Y für die

Gesamtzahl der Datensätze steht, die vor der Deduplizierung abgeglichen wurden. Dies weist darauf hin, dass doppelte Datensätze entfernt wurden, und bedeutet nicht, dass Daten fehlen.

Beispiel: Nur das letzte Protokollereignis für jeden eindeutigen Wert des Felds mit dem Namen **server** anzeigen

Das folgende Beispiel zeigt die Felder `timestamp`, `server`, `severity` und `message` nur für das letzte Ereignis für jeden eindeutigen Wert von `server` an.

```
fields @timestamp, server, severity, message
| sort @timestamp desc
| dedup server
```

Weitere Beispiele für CloudWatch Logs Insights-Abfragen finden Sie unter [Allgemeine Abfragen](#).

unmask

Verwenden Sie `unmask`, um den gesamten Inhalt eines Protokollereignisses anzuzeigen, bei dem einige Inhalte aufgrund einer Datenschutzrichtlinie maskiert sind. Um diesen Befehl zu verwenden, müssen Sie über die `logs:Unmask`-Berechtigung verfügen.

Weitere Informationen zum Datenschutz in Protokollgruppen finden Sie unter [Den Schutz vertraulicher Protokolldaten mit Maskierung unterstützen](#).

unnest

Wird verwendet `unnest`, um eine als Eingabe verwendete Liste zu vereinfachen, sodass mehrere Datensätze mit einem einzigen Datensatz für jedes Element in der Liste erstellt werden. Basierend auf der Anzahl der Elemente, die ein Feld enthält, verwirft dieser Befehl den aktuellen Datensatz und generiert neue Datensätze. Jeder Datensatz enthält `denunnested_field`, der ein Element darstellt. Alle anderen Felder stammen aus dem ursprünglichen Datensatz.

Die Eingabe für `unnest` ist `LIST`, die aus der `jsonParse` Funktion stammt. Weitere Informationen finden Sie unter [Strukturtypen](#). Alle anderen Typen `MAP`, wie z. B. `String` und `numbers`, werden als Liste mit einem Element behandelt `unnest`.

Befehlsstruktur

Das folgende Beispiel beschreibt das Format dieses Befehls.

```
unnest field into unnested_field
```

Beispielabfrage

Im folgenden Beispiel wird eine JSON-Objektzeichenfolge analysiert und eine Liste von Feldereignissen erweitert.

```
fields jsonParse(@message) as json_message
| unnest json_message.events into event
| display event.name
```

Das Protokollereignis für diese Beispielabfrage könnte eine JSON-Zeichenfolge wie folgt sein:

```
{
  "events": [
    {
      "name": "exception"
    },
    {
      "name": "user action"
    }
  ]
}
```

In diesem Fall erzeugt die Beispielabfrage zwei Datensätze im Abfrageergebnis, einen mit `event.name` als `exception` und einen mit `event.name` einer Benutzeraktion

Beispielabfrage

Im folgenden Beispiel wird eine Liste reduziert und anschließend Elemente herausgefiltert.

```
fields jsonParse(@message) as js
| unnest js.accounts into account
| filter account.type = "internal"
```

Beispielabfrage

Im folgenden Beispiel wird eine Liste zur Aggregation reduziert.

```
fields jsonParse(trimmedData) as accounts
| unnest accounts into account
| stats sum(account.droppedSpans) as n by account.accountId
| sort n desc
```

```
| limit 10
```

Nachschlagen

Wird verwendet `lookup`, um Ihre Abfrageergebnisse mit Referenzdaten aus einer Nachschlagetabelle anzureichern. Eine Nachschlagetabelle enthält CSV-Daten, die Sie in Amazon CloudWatch Logs hochladen. Wenn eine Abfrage ausgeführt wird, gleicht der `lookup` Befehl ein Feld in Ihren Protokollereignissen mit einem Feld in der Nachschlagetabelle ab und fügt die angegebenen Ausgabefelder an die Ergebnisse an.

Verwenden Sie Nachschlagetabellen für Szenarien zur Datenanreicherung, z. B. für die Zuordnung von Benutzer-IDs zu Benutzerdetails, von Produktcodes zu Produktinformationen oder von Fehlercodes zu Fehlerbeschreibungen.

Nachschlagetabellen erstellen und verwalten

Bevor Sie den `lookup` Befehl in einer Abfrage verwenden können, müssen Sie eine Nachschlagetabelle erstellen. Sie können Nachschlagetabellen über die CloudWatch Konsole oder mithilfe der Amazon CloudWatch Logs-API erstellen und verwalten.

Um eine Nachschlagetabelle (Konsole) zu erstellen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Einstellungen und dann die Registerkarte Protokolle aus.
3. Scrollen Sie zu „Nachschlagetabellen“ und wählen Sie „Verwalten“.
4. Wählen Sie Nachschlagetabelle erstellen aus.
5. Geben Sie einen Namen für die Nachschlagetabelle ein. Der Name darf nur alphanumerische Zeichen, Bindestriche und Unterstriche enthalten.
6. (Optional) Geben Sie eine Beschreibung ein.
7. Laden Sie eine CSV-Datei hoch. Die Datei muss eine Kopfzeile mit Spaltennamen enthalten, eine UTF-8 Kodierung verwenden und darf 10 MB nicht überschreiten.
8. (Optional) Geben Sie einen AWS KMS Schlüssel zum Verschlüsseln der Tabellendaten an.
9. Wählen Sie Erstellen aus.

Nachdem Sie eine Nachschlagetabelle erstellt haben, können Sie sie im CloudWatch Logs Insights-Abfrage-Editor anzeigen. Wählen Sie die Registerkarte Nachschlagetabellen, um die verfügbaren Tabellen und ihre Felder zu durchsuchen.

Um eine Nachschlagetabelle zu aktualisieren, wählen Sie die Tabelle aus und klicken Sie auf Aktionen, Aktualisieren. Laden Sie eine neue CSV-Datei hoch, um den gesamten vorhandenen Inhalt zu ersetzen. Um eine Nachschlagetabelle zu löschen, wählen Sie Aktionen, Löschen.

 Note

Sie können bis zu 100 Nachschlagetabellen pro Konto pro Konto erstellen AWS-Region. CSV-Dateien können bis zu 10 MB groß sein. Sie können Nachschlagetabellen auch mithilfe der Amazon CloudWatch Logs-API verwalten. Weitere Informationen finden Sie [CreateLookupTable](#) in der Amazon CloudWatch Logs API-Referenz.

 Note

Wenn die Nachschlagetabelle mit einem KMS-Schlüssel verschlüsselt ist, muss der Aufrufer über die `kms:Decrypt` Berechtigung für den Schlüssel (den KMS-Schlüssel, der zur Verschlüsselung der Nachschlagetabelle verwendet wird) verfügen, um die `StartQuery` API mit einer Abfrage zu verwenden, die auf diese Nachschlagetabelle verweist. Weitere Informationen finden Sie unter [Verschlüsseln Sie Nachschlagetabellen in CloudWatch Protokollen mit AWS Key Management Service](#).

Abfragesyntax für die Suche

Befehlsstruktur

Im Folgenden wird das Format dieses Befehls veranschaulicht.

```
lookup table match-fields output-mode output-field[,...]
```

Der Befehl verwendet die folgenden Argumente:

- *table*— Der Name der zu verwendenden Nachschlagetabelle.
- *match-fields*— Geben Sie ein oder mehrere Felder an, um Protokollereignisse mit der Nachschlagetabelle abzugleichen. Sie können eine der folgenden Formen verwenden:
 - *lookup-field* as *log-field* [*, ...*]*— as* Wird verwendet, wenn der Spaltenname der Nachschlagetabelle vom Namen des Protokollereignisfeldes abweicht. `ip_address` as

`srcAddr` vergleicht beispielsweise die `ip_address` Spalte in der Nachschlagetabelle mit dem `srcAddr` Feld in Ihren Protokollereignissen.

- ***lookup-field*** `[, ...]`— Wenn der Name des Protokollereignisfeldes mit dem Spaltennamen der Nachschlagetabelle identisch ist, können Sie den Feldnamen weglassen und direkt angeben. Ordnet beispielsweise `department`, `role` beide Spalten den gleichnamigen Protokollereignisfeldern zu.

Wenn mehrere Abgleichsfelder angegeben sind, muss eine Zeile in der Nachschlagetabelle allen Feldern entsprechen, um ein Ergebnis zu erhalten (UND-Logik).

- ***output-mode***— Gibt an, wie Ausgabefelder zu den Ergebnissen hinzugefügt werden. Nutzen Sie einen der Folgenden:
 - **OUTPUT**— Fügt die Ausgabefelder zu den Ergebnissen hinzu. Wenn ein Feld mit demselben Namen bereits im Protokollereignis vorhanden ist, wird es mit dem Wert der Nachschlagetabelle überschrieben. Wenn keine Übereinstimmung gefunden wird, wird das Feld auf Null gesetzt.
 - **OUTPUTNEW**— Fügt die Ausgabefelder nur dann zu den Ergebnissen hinzu, wenn das Feld nicht bereits im Protokollereignis vorhanden ist. Wenn das Feld bereits einen Wert hat, wird der ursprüngliche Wert beibehalten. Wenn keine Übereinstimmung gefunden wird, bleibt das Feld unverändert.
- ***output-field***— Ein oder mehrere Felder aus der Nachschlagetabelle, die den Ergebnissen hinzugefügt werden sollen.

Beispiel: Reichern Sie Protokollereignisse mit Benutzerdetails an

Angenommen, Sie haben eine Protokollgruppe mit Ereignissen, die ein `id` Feld enthalten, und eine Nachschlagetabelle `user_data` mit den Spalten `id`, `name`, `email`, und `department`. Die folgende Abfrage bereichert jedes Protokollereignis mit dem Namen, der E-Mail-Adresse und der Abteilung des Benutzers aus der Nachschlagetabelle.

```
fields action, status, name, email, department
| lookup user_data id OUTPUT name, email, department
```

Beispiel: Verwenden Sie Lookup mit Aggregation

Sie können Lookup-Ausgabefelder mit Aggregationsfunktionen verwenden. Die folgende Abfrage reichert Protokollereignisse mit Benutzerdetails an und zählt dann die Ereignisse, gruppiert nach E-Mail-Adresse.

```
fields user_id, action, username, email, department
| lookup user_data user_id OUTPUT username, email, department
| stats count(*) by email
```

Beispiel: Verwenden Sie die Suche mit Filter

Sie können Ergebnisse auf der Grundlage von Feldern filtern, die bei der Suche zurückgegeben wurden. Mit der folgenden Abfrage werden Protokollereignisse erweitert und anschließend gefiltert, sodass nur Ereignisse aus einer bestimmten Abteilung angezeigt werden.

```
fields user_id, action
| lookup user_data user_id OUTPUT username, email, department
| filter department = "Engineering"
```

Beispiel: Verwenden Sie OUTPUTNEW, um die Daten anzureichern, ohne sie zu überschreiben

Wenn Ihre Protokollereignisse bereits ein hostname Feld enthalten, dieses aber manchmal leer ist, können Sie es verwenden, OUTPUTNEW um fehlende Werte auszufüllen, ohne die vorhandenen zu überschreiben.

```
fields srcAddr, hostname
| lookup known_hosts ip_address as srcAddr OUTPUTNEW hostname, region
```

Beispiel: Verwenden Sie die Suche mit mehreren Übereinstimmungsfeldern

Sie können in mehr als einem Feld einen Abgleich vornehmen. Die folgende Abfrage gleicht srcAddr sowohl als auch mit der dstPort Nachschlagetabelle ab, um bekannte Netzwerkdienste zu identifizieren.

```
fields @timestamp, srcAddr, dstAddr, dstPort
| lookup network_services ip_address as srcAddr, port as dstPort OUTPUT service_name,
  owner
| filter ispresent(service_name)
```

Beispiel: Verwenden Sie die Suche mit übereinstimmenden Feldnamen

Wenn die Feldnamen Ihrer Protokollereignisse genau mit den Spaltennamen der Nachschlagetabelle übereinstimmen, können Sie das as Schlüsselwort weglassen. In der folgenden Abfrage werden beide department role Felder direkt mit der Nachschlagetabelle verglichen.

```
fields @timestamp, department, role
| lookup employees department, role OUTPUT office, manager
```

join

Kombiniert Protokollereignisse aus einer Quellprotokollgruppe mit Ereignissen aus einer anderen Protokollgruppe oder einem Abfrageergebnis, das auf einem passenden Feld basiert.

Verwenden Sie den `join` Befehl, um verwandte Protokollereignisse aus verschiedenen Quellen wie Protokollgruppen zu korrelieren. Verwenden Sie dabei Schlüssel, die allen gemeinsam sind, z. B. übereinstimmende Anforderungskennungen oder Transaktions-IDs.

Syntax

```
join type=<join_type> left=<left_alias> right=<right_alias>
  where <left_alias>.<field>=<right_alias>.<field>
  (SOURCE <right_log_group>)
```

Parameters

- `<right_log_group>`— Die sekundäre Datenquelle, mit der eine Verbindung hergestellt werden soll.
- `<left_alias>` und `<right_alias>` — Aliase, um Felder von den linken (primären) und rechten (sekundären) Datenquellen zu unterscheiden.
- `where <field>`— Gibt das Feld an, das als Join-Schlüssel verwendet wird. Das Feld muss in beiden Datenquellen vorhanden sein.
- `type=<join_type>(optional)` — Gibt den Join-Typ an. Folgende sind gültige Werte:
 - `inner(Standard)` — Gibt nur übereinstimmende Datensätze zurück
 - `left`— Gibt alle Datensätze aus der primären Datenquelle und übereinstimmende Datensätze aus der sekundären Datenquelle zurück

Beispiele

Example Beispiel 1: API-Gateway-Anfragen mit Lambda-Ausführungsprotokollen korrelieren

Dieses Beispiel zeigt, wie API Gateway Gateway-Zugriffsprotokolle mit Lambda-Funktionsprotokollen verknüpft werden, um eingehende Anfragen mit ihrer Backend-Verarbeitung zu korrelieren. Dies ist

nützlich, um End-to-End-Anforderungsabläufe zu beheben und zu ermitteln, welche Lambda-Aufrufe bestimmten API-Anfragen entsprechen.

```
filter status >= 500
| join type=inner left=api right=lambda
  where api.requestId=lambda.requestId
  (SOURCE '/aws/lambda/my-function')
| fields api.requestId, api.status, api.latency, lambda.duration, lambda.memoryUsed
| sort api.latency desc
```

Diese Abfrage:

1. Fragt API-Gateway-Zugriffsprotokolle ab und filtert sie nach Serverfehlern (Status >= 500)
2. Verbindet sich mit Lambda-Funktionsprotokollen unter Verwendung des requestId Felds, das in beiden Protokollquellen erscheint
3. Verwendet Aliase (apiundlambda), um Felder aus den einzelnen Quellen zu unterscheiden
4. Gibt kombinierte Informationen zurück, die die API-Latenz zusammen mit der Lambda-Ausführungsdauer und der Speichernutzung zeigen
5. Sortiert die Ergebnisse nach API-Latenz, um die langsamsten Anfragen zu identifizieren

Example Beispiel 2: Verfolgen Sie verteilte Transaktionen über Microservices hinweg

Beim Debuggen von Problemen in einer Microservices-Architektur müssen Sie häufig eine Transaktion über mehrere Dienste hinweg verfolgen. Dieses Beispiel zeigt, wie Logs von zwei verschiedenen Diensten mithilfe einer gemeinsamen Transaktions-ID verknüpft werden.

```
filter eventType = "ORDER_CREATED"
| join type=left left=order right=payment
  where order.transactionId=payment.transactionId
  (SOURCE '/aws/lambda/payment-service')
| filter payment.eventType = "PAYMENT_PROCESSED" or !ispresent(payment.eventType)
| fields order.transactionId, order.orderId, order.customerId,
  payment.paymentStatus, payment.amount
| filter payment.paymentStatus != "SUCCESS" or !ispresent(payment.paymentStatus)
```

Diese Abfrage:

1. Beginnt mit Ereignissen zur Auftragserstellung vom Bestellservice

2. Benutzt `aleft join`, um alle Bestellungen einzubeziehen, auch solche ohne übereinstimmende Zahlungsbelege
3. Verbindet sich mithilfe des gemeinsam genutzten `transactionId` Felds mit Ereignissen zur Zahlungsabwicklung
4. Filtert die Endergebnisse so, dass nur Bestellungen mit fehlgeschlagenen Zahlungen oder fehlenden Zahlungsdatensätzen angezeigt werden

Die linke Verknüpfung ist hier wichtig, da sie sicherstellt, dass Sie Bestellungen sehen, die zwar erstellt wurden, aber nie ein entsprechendes Zahlungsereignis hatten, was auf einen Systemausfall hindeuten könnte.

Behavior

- Die primäre Datenquelle (linke Seite) wird zuerst verarbeitet.
- Die sekundäre Datenquelle wird anhand des angegebenen Join-Schlüssels ausgewertet und abgeglichen.
- Der Abgleich erfolgt mithilfe eines Gleichheitsvergleichs für das Join-Feld.
- Bei linken Verknüpfungen werden Datensätze ohne Übereinstimmung aus der primären Datenquelle mit Nullwerten für sekundäre Felder beibehalten.

Hinweise und Einschränkungen

- Es werden nur Gleichheitsbedingungen (=) unterstützt.
- Pro Abfrage wird nur ein Join-Befehl unterstützt.
- Verbindungsschlüssel müssen in beiden Datenquellen vorhanden sein und einen kompatiblen Typ haben.
- Abfragen, die Join verwenden, können mehr Daten scannen und höhere Kosten verursachen.
- Die Anzahl der eindeutigen Schlüsselwerte in der sekundären Datenquelle ist auf 50.000 begrenzt, um die Abfrageleistung sicherzustellen.
- Unterabfragen auf der rechten Seite des Joins werden nicht unterstützt.

Zugehörige Befehle

- [Felder](#)
- [Filter](#)
- [analysieren](#)
- [Statistiken](#)
- [sortieren](#)
- [limit](#)

Unterabfragen

Eine Unterabfrage ist eine verschachtelte Logs Insights-Abfrage, die als Eingabe für eine andere Abfrage verwendet werden kann. Unterabfragen können verwendet werden, um Zwischenergebnismengen abzuleiten, die dann von nachfolgenden Befehlen verwendet werden.

Syntax

Unterabfrage im Filter

```
filter <field> in (  
    <subquery>  
)
```

Parameters

- **<subquery>**— Eine gültige Logs Insights-Abfrage, die eine Ergebnismenge zurückgibt. Die Unterabfrage muss Felder erzeugen, auf die von der äußeren Abfrage verwiesen wird.

Beispiele

Example Beispiel 1: Suchen Sie nach Anfragen, bei denen Fehler in nachgelagerten Diensten aufgetreten sind

Dieses Beispiel zeigt, wie Sie mithilfe einer Unterabfrage Anfragen in Ihrem Hauptdienst identifizieren, die zu Fehlern in einem nachgelagerten Dienst geführt haben. Dies ist nützlich für die Behebung kaskadierender Fehler in verteilten Systemen.

```
filter requestId in (
  SOURCE '/aws/lambda/database-service'
  | filter errorType = "DatabaseConnectionTimeout"
  | fields requestId
)
| fields @timestamp, requestId, endpoint, userId, responseTime
| sort @timestamp desc
```

Diese Abfrage:

1. Die Unterabfrage findet alle `requestId` Werte aus dem Datenbankdienst, bei dem es zu Verbindungs-Timeouts kam
2. Die äußere Abfrage filtert die Protokolle Ihres Hauptdienstes, sodass nur Anfragen angezeigt werden, die diesen fehleranfälligen Anforderungs-IDs entsprechen
3. Die Ergebnisse zeigen den vollständigen Kontext der Anfragen, die im Downstream fehlgeschlagen sind, einschließlich der betroffenen Endgeräte und Benutzer

Dieses Muster hilft Ihnen, die Auswirkungen von Downstream-Ausfällen im Upstream-Bereich zu verstehen.

Example Beispiel 2: Identifizieren Sie häufig fehlgeschlagene Anfragen für gezielte Untersuchungen

Dieses Beispiel zeigt, wie eine Unterabfrage mit Aggregation verwendet wird, um Anfragen zu finden, die wiederholt fehlschlagen, was häufig eher auf systematische Probleme als auf vorübergehende Fehler hindeutet.

```
filter requestId in (
  SOURCE '/aws/lambda/payment-processor'
  | filter status = "FAILED"
  | stats count(*) as failureCount by requestId
  | filter failureCount > 3
  | fields requestId
)
| fields @timestamp, requestId, customerId, amount, failureReason
| sort @timestamp asc
```

Diese Abfrage:

1. Die Unterabfrage aggregiert fehlgeschlagene Zahlungsversuche und identifiziert Anforderungs-IDs, die mehr als dreimal fehlgeschlagen sind

2. Die äußere Abfrage ruft alle Protokollereignisse für diese problematischen Anforderungs-IDs ab
3. Die Ergebnisse sind chronologisch sortiert, um den Verlauf der Wiederholungsversuche aufzuzeigen

Dies hilft bei der Unterscheidung zwischen vorübergehenden Ausfällen (einmaliges Auftreten) und anhaltenden Problemen (mehrere Ausfälle), die eingehendere Untersuchungen erfordern.

Behavior

- Unterabfragen werden unabhängig von der äußeren Abfrage ausgeführt.
- Die Ergebnisse werden materialisiert, bevor sie von der äußeren Abfrage verarbeitet werden.
- Nur Felder, die explizit in der Unterabfrage ausgewählt wurden, sind für die äußere Abfrage verfügbar.

Hinweise und Einschränkungen

- Unterabfragen müssen Felder zurückgeben, auf die in der äußeren Abfrage verwiesen wird.
- Verschachtelte Unterabfragen werden nicht unterstützt.
- Unterabfragen können die Ausführungszeit und die Kosten für die Ausführung von Abfragen erhöhen.
- Korrelierte Unterabfragen werden nicht unterstützt.
- Die Ausführung innerer Abfragen ist auf 30 Sekunden begrenzt.

Zugehörige Befehle

- [Felder](#)
- [Filter](#)
- [analysieren](#)
- [Statistiken](#)
- [sortieren](#)
- [limit](#)

Boolesche, Vergleichs-, numerische, Datetime- und andere Funktionen

CloudWatch Logs Insights unterstützt viele andere Operationen und Funktionen in Abfragen, wie in den folgenden Abschnitten erläutert.

Themen

- [Arithmetische Operatoren](#)
- [Boolesche Operatoren](#)
- [Vergleichsoperatoren](#)
- [Numerische Operatoren](#)
- [Strukturtypen](#)
- [Datum-/Uhrzeit-Funktionen](#)
- [Allgemeine Funktionen](#)
- [JSON-Funktionen](#)
- [IP-Adressenzeichenfolgen](#)
- [Zeichenfolgenfunktionen](#)

Arithmetische Operatoren

Arithmetische Operationen akzeptieren numerische Datentypen als Argumente und liefern numerische Ergebnisse. Sie können arithmetische Operationen in den Befehlen `filter` und `fields` sowie als Argumente für andere Funktionen verwenden.

Operation	Description
$a + b$	Addition
$a - b$	Subtraktion
$a * b$	Multiplikation
a / b	Division
$a ^ b$	Potenzierung. $2 ^ 3$ gibt 8 zurück
$a \% b$	Rest oder Modulus. $10 \% 3$ gibt 1 zurück

Boolesche Operatoren

Sie können die booleschen Operatoren **and**, **or**, und **not** verwenden.

Note

Verwenden Sie boolesche Operatoren nur in Funktionen, die den Wert von WAHR oder FALSCH ausgeben.

Vergleichsoperatoren

Vergleichsoperationen akzeptieren alle Datentypen als Argumente und liefern ein boolesches Ergebnis. Sie können Vergleichsoperationen im Befehl `filter` sowie als Argumente für andere Funktionen verwenden.

Operator	Description
=	Gleich
!=	Ungleich
<	Kleiner als
>	Größer als
<=	Kleiner als oder gleich
>=	Größer als oder gleich

Numerische Operatoren

Numerische Operationen akzeptieren numerische Datentypen als Argumente und liefern numerische Ergebnisse. Sie können numerische Operationen in den Befehlen `filter` und `fields` sowie als Argumente für andere Funktionen verwenden.

Operation	Ergebnistyp	Description
<code>abs(a: number)</code>	number	Absoluter Wert

Operation	Ergebnistyp	Description
<code>ceil(a: number)</code>	number	Aufrunden (die kleinste ganze Zahl, die größer ist als der Wert von a).
<code>floor(a: number)</code>	number	Abrunden (die größte ganze Zahl, die kleiner ist als der Wert von a).
<code>greatest(a: number, ...numbers: number[])</code>	number	Liefert den größten Wert.
<code>least(a: number, ...numbers: number[])</code>	number	Liefert den kleinsten Wert.
<code>log(a: number)</code>	number	Natürlicher Logarithmus
<code>round(a: number [, d: number])</code>	number	Rundet den Wert von a. Rundet mit einem Argument auf die nächste Ganzzahl. Rundet bei zwei Argumenten auf d Dezimalstellen.
<code>sqrt(a: number)</code>	number	Quadratwurzel
<code>haversine(lat1: number, lon1: number, lat2: number, lon2: number)</code>	number	Berechnet die Großkreis entfernung in Kilometern zwischen zwei geographischen Punkten, die durch Breiten- und Längengrad in Grad angegeben sind.

Strukturtypen

Eine Karte oder Liste ist ein Strukturtyp in CloudWatch Logs Insights, mit dem Sie auf Attribute zugreifen und diese für Abfragen verwenden können.

Beispiel: Um eine Karte oder Liste abzurufen

Wird verwendet `jsonParse`, um ein Feld, das eine JSON-Zeichenfolge ist, in eine Map oder eine Liste zu parsen.

```
fields jsonParse(@message) as json_message
```

Beispiel: Um auf Attribute zuzugreifen

Verwenden Sie den Punktzugriffsoperator (`map.attribute`), um auf Elemente in einer Map zuzugreifen. Wenn ein Attribut in einer Map Sonderzeichen enthält, verwenden Sie Backticks, um den Attributnamen einzuschließen (`map.attributes`). ``special.char``.

```
fields jsonParse(@message) as json_message  
| stats count() by json_message.status_code
```

Verwenden Sie den Klammer-Zugriffsoperator (`list [index]`), um ein Element an einer bestimmten Position in der Liste abzurufen.

```
fields jsonParse(@message) as json_message  
| filter json_message.users[1].action = "PutData"
```

Umbrechen Sie Sonderzeichen in Backticks (```), wenn Sonderzeichen im Schlüsselnamen enthalten sind.

```
fields jsonParse(@message) as json_message  
| filter json_message.`user.id` = "123"
```

Beispiel: leere Ergebnisse

Karten und Listen werden für Zeichenketten-, Zahlen- und Datetime-Funktionen als Null behandelt.

```
fields jsonParse(@message) as json_message  
| display toupper(json_message)
```

Der Vergleich von Karte und Liste mit allen anderen Feldern führt zu `false`.

 Note

Die Verwendung von Map und List in `deduppattern`, `sort`, und `wildcardstats` wird nicht unterstützt.

Datum-/Uhrzeit-Funktionen

Datum-/Uhrzeit-Funktionen

Sie können die Datums-/Uhrzeit-Funktionen in den Befehlen `fields` und `filter` sowie als Argumente für andere Funktionen verwenden. Mit diesen Funktionen können Sie Zeiträume für Abfragen mit Aggregationsfunktionen anlegen. Verwenden Sie Zeiträume, die aus einer Zahl und einem der folgenden Werte bestehen:

- `ms` für Millisekunden
- `s` für Sekunden
- `m` für Minuten
- `h` stundenlang

Zum Beispiel steht `10m` für 10 Minuten und `1h` ist 1 Stunde.

 Note

Verwenden Sie die am besten geeignete Zeiteinheit für Ihre Datetime-Funktion. CloudWatch Logs begrenzt Ihre Anfrage auf die von Ihnen gewählte Zeiteinheit. Beispielsweise wird für jede Anfrage, die verwendet, eine Obergrenze von 60 als Maximalwert festgelegt. Wenn Sie also angeben `bin(300s)`, implementiert CloudWatch Logs dies tatsächlich als 60 Sekunden, da 60 die Anzahl der Sekunden in einer Minute ist, sodass CloudWatch Logs keine höhere Zahl als 60 verwendet. Um einen 5-Minuten-Bucket zu erstellen, verwenden Sie `bin(5m)` stattdessen.

Die Obergrenze für `ms` ist 1000, die Obergrenzen für `s` und `m` sind 60 und die Obergrenze für `h` ist 24.

Die folgende Tabelle enthält eine Liste der verschiedenen Datetime-Funktionen, die Sie in Ihren Abfragebefehlen verwenden können. Die Tabelle listet den Ergebnistyp jeder Funktion auf und enthält zu jeder Funktion eine Beschreibung.

 Tip

Wenn Sie einen Abfragebefehl erstellen, können Sie mit der Zeitintervallauswahl einen Zeitraum für Ihre Abfrage festlegen. Sie können beispielsweise Intervalle von 5 bis 30 Minuten, Intervalle von 1, 3 und 12 Stunden oder einen benutzerdefinierten Zeitrahmen festlegen. Sie können auch Zeiträume zwischen bestimmten Daten festlegen.

Funktion	Ergebnistyp	Description
<code>bin(period: Period)</code>	Zeitstempel	Rundet den Wert von <code>@timestamp</code> auf den angegebenen Zeitraum und kürzt ihn dann. <code>bin(5m)</code> rundet beispielsweise den Wert von <code>@timestamp</code> auf die nächstgelegenen fünf Minuten. So können mehrere Protokolleinträge in einer Abfrage gruppiert werden. Im folgenden Beispiel wird die Anzahl von Ausnahmen pro Stunde zurückgegeben: <pre>filter @message like /Exception/ stats count(*) as exceptionCount by bin(1h) sort exceptionCount desc</pre> Für die Funktion <code>bin</code> werden folgende Zeiteinheiten und Abkürzungen unterstützt. Alle Einheiten und Abkürzungen, die mehr als ein Zeichen enthalten, können durch Hinzufügen von „s“ pluralisiert werden. Somit kann sowohl <code>hr</code> als auch <code>hrs</code> verwendet werden, um Stunden anzugeben.

Funktion	Ergebnistyp	Description
		• <code>millisecond ms msec</code> • <code>second s sec</code> • <code>minute m min</code> • <code>hour h hr</code> • <code>day d</code> • <code>week w</code> • <code>month mo mon</code> • <code>quarter q qtr</code> • <code>year y yr</code>
<code>datefloor(timestamp: Timestamp, period: Period)</code>	Zeitstempel	Kürzt den Zeitstempel auf den angegebenen Zeitraum. Zum Beispiel kürzt <code>datefloor(@timestamp, 1h)</code> alle Werte von <code>@timestamp</code> auf die letzte volle Stunde.
<code>dateceil(timestamp: Timestamp, period: Period)</code>	Zeitstempel	Rundet den Zeitstempel auf den angegebenen Zeitraum auf und kürzt ihn dann. Zum Beispiel kürzt <code>dateceil(@timestamp, 1h)</code> alle Werte von <code>@timestamp</code> auf die nächste volle Stunde.
<code>fromMillis(fieldName: number)</code>	Zeitstempel	Interpretiert das Eingabefeld als die Anzahl der Millisekunden seit der Unix-Epoche und konvertiert es in einen Zeitstempel.
<code>toMillis(fieldName: Timestamp)</code>	number	Konvertiert den im benannten Feld gefundene n Zeitstempel in eine Zahl, die die Millisekunden seit der Unix-Epoche darstellt. Beispiel: <code>toMillis(@timestamp)</code> konvertiert den Zeitstempel <code>2022-01-14T13:18:03.000-08:00</code> zu <code>1642195111000</code> .

Funktion	Ergebnistyp	Description
now()	number	Gibt die Uhrzeit in Epochensekunden zurück, zu der die Abfrageverarbeitung gestartet wurde. Diese Funktion akzeptiert keine Argumente. Sie können dies verwenden, um Ihre Abfrageergebnisse nach der aktuellen Uhrzeit zu filtern. Die folgende Abfrage gibt beispielsweise alle 4xx-Fehler der letzten zwei Stunden zurück: <pre>parse @message "Status Code: *;" as statusCode\n filter statusCode >= 400 and statusCode <= 499 \n filter toMillis(@timestamp) >= (now() * 1000 - 7200000)</pre> Das folgende Beispiel gibt alle Protokolleinträge der letzten fünf Stunden zurück, die entweder das Wort error oder enthalten failure <pre>fields @timestamp, @message filter @message like /(?(i)(error failure))/ filter toMillis(@timestamp) >= (now() * 1000 - 18000000)</pre>

Note

Derzeit unterstützt CloudWatch Logs Insights das Filtern von Protokollen mit menschenlesbaren Zeitstempeln nicht.

Allgemeine Funktionen

Allgemeine Funktionen

Sie können allgemeine Funktionen in den Befehlen `fields` und `filter` sowie als Argumente für andere Funktionen verwenden.

Funktion	Ergebnistyp	Description
<code>ispresent(fieldName: LogField)</code>	Boolesch	Gibt <code>true</code> zurück, wenn das Feld existiert.
<code>coalesce(fieldName: LogField, ...fieldNames: LogField[])</code>	LogField	Liefert den ersten Nicht-Null-Wert aus der Liste.
<code>case(cond1, val1, cond2, val2, ..., [default])</code>	LogField	Wertet die Bedingungen der Reihe nach aus und gibt den Wert für die erste wahre Bedingung zurück. Wenn keine Bedingung wahr ist und ein Standardwert angegeben ist, wird der Standardwert zurückgegeben. Unterstützt bis zu 10 Zweige.

JSON-Funktionen

JSON-Funktionen

Verwenden Sie JSON-Funktionen in den `filter` Befehlen `fields` und und als Argumente für andere Funktionen.

Funktion	Ergebnistyp	Description
<code>jsonParse(fieldName: string)</code>	Karte Liste Leer	Gibt eine Map oder Liste zurück, wenn es sich bei der Eingabe um eine Zeichenkettendarstellung eines JSON-Objekts oder eines JSON-Arrays handelt. Gibt einen leeren Wert zurück, wenn es sich bei

Funktion	Ergebnistyp	Description
		der Eingabe nicht um eine der Repräsentationen handelt.
<code>jsonStringify(fieldName: Map List)</code>	Zeichenfolge	Gibt eine JSON-Zeichenfolge aus einer Karte oder Listendaten zurück.

IP-Adressenzeichenfolgen

IP-Adressenzeichenfolgen

Sie können IP-Adressen-Zeichenfolgenfunktionen in den Befehlen `filter` und `fields` sowie als Argumente für andere Funktionen verwenden.

Funktion	Ergebnistyp	Description
<code>isValidIp(fieldName: string)</code>	boolesch	Gibt <code>true</code> zurück, wenn das Feld eine gültige IPv4- oder IPv6-Adresse ist.
<code>isValidIPv4(fieldName: string)</code>	boolesch	Gibt <code>true</code> zurück, wenn das Feld eine gültige IPv4-Adresse ist.
<code>isValidIPv6(fieldName: string)</code>	boolesch	Gibt <code>true</code> zurück, wenn das Feld eine gültige IPv6-Adresse ist.
<code>isIpInSubnet(fieldName: string, subnet: string)</code>	boolesch	Gibt <code>true</code> zurück, wenn das Feld eine gültige IPv4- oder IPv6-Adresse innerhalb des angegebenen IPv4- oder IPv6-Subnetzes ist. Verwenden Sie bei der Angabe des Subnetzes die CIDR-Notation wie <code>192.0.2.0/24</code> oder <code>2001:db8::/32</code> , wobei <code>192.0.2.0</code> oder <code>2001:db8::</code> der Anfang des CIDR-Blocks ist.
<code>isIPv4InSubnet(fieldName: string, subnet: string)</code>	boolesch	Gibt <code>true</code> zurück, wenn das Feld eine gültige IPv4-Adresse innerhalb des angegebenen v4-Subnetzes ist. Verwenden Sie bei der Angabe des Subnetzes die CIDR-Notation wie

Funktion	Ergebnistyp	Description
		192.0.2.0/24 , wobei 192.0.2.0 der Anfang des CIDR-Blocks ist.
<code>isIpv6InSubnet(fieldName: string, subnet: string)</code>	boolesch	Gibt <code>true</code> zurück, wenn das Feld eine gültige IPv6-Adresse innerhalb des angegebenen v6-Subnetzes ist. Verwenden Sie bei der Angabe des Subnetzes die CIDR-Notation wie <code>2001:db8::/32</code> , wobei <code>2001:db8::</code> der Anfang des CIDR-Blocks ist.

Zeichenfolgenfunktionen

Zeichenfolgenfunktionen

Sie können Zeichenfolgenfunktionen in den Befehlen `fields` und `filter` sowie als Argumente für andere Funktionen verwenden.

Funktion	Ergebnistyp	Description
<code>isempty(fieldName: string)</code>	Zahl	Gibt 1 zurück, wenn das Feld fehlt oder eine leere Zeichenkette ist.
<code>isblank(fieldName: string)</code>	Zahl	Gibt 1 zurück, wenn das Feld fehlt, eine leere Zeichenkette ist oder nur Leerzeichen enthält.
<code>concat(str: string, ...strings: string[])</code>	Zeichenfolge	Verkettet die Zeichenketten.
<code>ltrim(str: string)</code> <code>ltrim(str: string, trimChars: string)</code>	Zeichenfolge	Wenn die Funktion kein zweites Argument hat, entfernt sie die Whitespaces von der linken Seite der Zeichenfolge. Wenn die Funktion ein zweites

Funktion	Ergebnistyp	Description
		Zeichenfolgen-Argument hat, entfernt sie keine Whitespaces. Stattdessen entfernt sie die Zeichen in <code>trimChars</code> links von <code>str</code> . Beispielsweise gibt <code>ltrim("xyZxyfooxyZ", "xyZ")</code> <code>"fooxyZ"</code> zurück.
<code>rtrim(str: string)</code> <code>rtrim(str: string, trimChars: string)</code>	Zeichenfolge	Wenn die Funktion kein zweites Argument hat, entfernt sie die Whitespaces von der rechten Seite der Zeichenfolge. Wenn die Funktion ein zweites Zeichenfolgen-Argument hat, entfernt sie keine Whitespaces. Stattdessen entfernt sie die Zeichen von <code>trimChars</code> rechts von <code>str</code> . Beispielsweise gibt <code>rtrim("xyZfooxyxyZ", "xyZ")</code> <code>"xyZfoo"</code> zurück.

Funktion	Ergebnistyp	Description
<code>trim(str: string)</code> <code>trim(str: string, trimChars: string)</code>	Zeichenfolge	Wenn die Funktion kein zweites Argument hat, entfernt sie die Whitespaces von beiden Seiten der Zeichenfolge. Wenn die Funktion ein zweites Zeichenfolgen-Argument hat, entfernt sie keine Whitespaces. Stattdessen entfernt sie die Zeichen aus <code>trimChars</code> von beiden Seiten von <code>str</code> . Beispielsweise gibt <code>trim("xyZxyfooxyxyZ", "xyZ")</code> <code>"foo"</code> zurück.
<code>strlen(str: string)</code>	number	Liefert die Länge der Zeichenkette in Unicode-Codpunkten.
<code>toupper(str: string)</code>	Zeichenfolge	Konvertiert die Zeichenkette in Großbuchstaben.
<code>tolower(str: string)</code>	Zeichenfolge	Konvertiert die Zeichenkette in Kleinbuchstaben.

Funktion	Ergebnistyp	Description
<pre>substr(str: string, startIndex: number)</pre> <pre>substr(str: string, startIndex: number, length: number)</pre>	Zeichenfolge	Gibt eine Teilzeichenkette aus dem durch das Zahlenargument angegebenen Index bis zum Ende der Zeichenkette zurück. Wenn die Funktion ein zweites Zahlenargument hat, enthält sie die Länge der abzurufenden Teilzeichenkette. Beispielsweise gibt <code>substr("xyZfooxyZ", 3, 3)</code> "foo" zurück.
<pre>replace(fieldName: string, searchValue: string, replaceValue: string)</pre>	Zeichenfolge	Ersetzt alle Instances von <code>searchValue</code> in <code>fieldName: string</code> mit <code>replaceValue</code>. Beispiel: Die Funktion <code>replace(logGroup, "smoke_test", "Smoke")</code> sucht nach Protokollereignissen, bei denen das Feld <code>logGroup</code> den Zeichenfolgenwert <code>smoke_test</code> enthält und den Wert durch die Zeichenfolge <code>Smoke</code> ersetzt.
<pre>regex_replace(fieldName: string, pattern: string, replacement: string)</pre>	Zeichenfolge	Ersetzt alle Teilzeichenfolgen, die dem regulären Ausdruck <code>pattern</code> entsprechen durch <code>replacement</code> . Verwendet die RE2-Regex-Syntax.

Funktion	Ergebnistyp	Description
<code>strcontains(str: string, searchValue: string)</code>	number	Gibt 1 zurück, wenn <code>str</code> <code>searchValue</code> enthält; ansonsten 0.
<code>startsWith(str: string, searchValue: string)</code>	number	Gibt 1 zurück, wenn mit <code>str</code> <code>searchValue</code> beginnt, andernfalls 0.
<code>endsWith(str: string, searchValue: string)</code>	number	Gibt 1 zurück, wenn es mit <code>str</code> <code>searchValue</code> endet, andernfalls 0.
<code>urlencode(str: string)</code>	Zeichenfolge	URL-encodes die Zeichenfolge.
<code>urldecode(str: string)</code>	Zeichenfolge	URL-decodes die Schnur.
<code>base64encode(str: string)</code>	Zeichenfolge	Base64-encodes die Schnur.
<code>base64decode(str: string)</code>	Zeichenfolge	Base64-decodes die Schnur.

Felder, die Sonderzeichen enthalten

Wenn ein Feld außer dem @ Symbol oder dem Punkt (.) keine alphanumerischen Zeichen enthält, müssen Sie das Feld mit Backtick-Zeichen (` ) umgeben. ` Zum Beispiel muss das Feld `foo-bar` in Backticks eingeschlossen werden (``foo-bar``), weil es ein nicht alphanumerisches Zeichen, den Bindestrich (-), enthält.

Aliasse und Kommentare in Abfragen verwenden

Erstellen Sie Abfragen, die Aliase enthalten. Verwenden Sie Aliase, um Protokollfelder umzubenennen oder Werte in Felder zu extrahieren. Benutze das Schlüsselwort `as` einem Protokollfeld oder Ergebnis einen Alias zu geben. Sie können in einer Abfrage mehrere Alias verwenden. In den folgenden Befehlen können Sie Aliase verwenden:

- `fields`
- `parse`

- `sort`
- `stats`

In den folgenden Beispielen wird gezeigt, wie Sie Abfragen erstellen, die Aliase enthalten.

Beispiel

Die Abfrage enthält einen Alias im `fields`-Befehl.

```
fields @timestamp, @message, accountId as ID
| sort @timestamp desc
| limit 20
```

Die Abfrage gibt die Werte für die Felder zurück `@timestamp`, `@message`, und `accountId` aus. Die Ergebnisse sind in absteigender Reihenfolge sortiert und auf 20 begrenzt. Die Werte für `accountId` sind unter dem Alias aufgeführt `ID` aus.

Beispiel

Die Abfrage enthält Aliase im `sort` und `stats`-Befehle.

```
stats count(*) by duration as time
| sort time desc
```

Die Abfrage zählt die Anzahl der Male des Feldes `duration` tritt in der Protokollgruppe auf und sortiert die Ergebnisse in absteigender Reihenfolge. Die Werte für `duration` sind unter dem Alias aufgeführt `time` aus.

Kommentare verwenden

CloudWatch Logs Insights unterstützt Kommentare in Abfragen. Benutze das Hash-Zeichen (`#`) um Kommentare auszurichten. Sie können Kommentare verwenden, um Zeilen in Abfragen oder Dokumentabfragen zu ignorieren.

Beispiel: Abfrage

Wenn die folgende Abfrage ausgeführt wird, wird die zweite Zeile ignoriert.

```
fields @timestamp, @message, accountId
# | filter accountId not like "7983124201998"
```

```
| sort @timestamp desc  
| limit 20
```

Erste Schritte mit Logs Insights QL: Query tutorials

Die folgenden Abschnitte enthalten Beispiel-Tutorials für Abfragen, die Ihnen den Einstieg in Logs Insights QL erleichtern.

Themen

- [Tutorial: Ausführen und Ändern einer Beispielabfrage](#)
- [Tutorial: Ausführen einer Abfrage mit einer Aggregationsfunktion](#)
- [Tutorial: Ausführen einer Abfrage, die eine nach Protokollfeldern gruppierte Visualisierung erzeugt](#)
- [Tutorial: Ausführen einer Abfrage, die eine Visualisierung von Zeitreihen erzeugt](#)

Tutorial: Ausführen und Ändern einer Beispielabfrage

Das folgende Tutorial hilft Ihnen bei den ersten Schritten mit CloudWatch Logs Insights. Sie führen eine Beispielabfrage in Logs Insights QL aus und sehen dann, wie Sie sie ändern und erneut ausführen können.

Um eine Abfrage ausführen zu können, müssen Sie bereits CloudWatch Protokolle in Logs gespeichert haben. Wenn Sie Logs bereits verwenden und CloudWatch Log-Gruppen und Log-Streams eingerichtet haben, können Sie loslegen. Möglicherweise verfügen Sie auch bereits über Protokolle AWS CloudTrail, wenn Sie Dienste wie Amazon Route 53 oder Amazon VPC verwenden und Protokolle von diesen Diensten so eingerichtet haben, dass sie in CloudWatch Logs gespeichert werden. Weitere Informationen zum Senden von Protokollen an CloudWatch Logs finden Sie unter [Erste Schritte mit CloudWatch Logs](#).

Abfragen in CloudWatch Logs Insights geben entweder eine Reihe von Feldern aus Protokollereignissen oder das Ergebnis einer mathematischen Aggregation oder eines anderen Vorgangs zurück, der mit Protokollereignissen ausgeführt wurde. Dieses Tutorial demonstriert eine Abfrage, die eine Liste von Protokollereignissen zurückgibt.

Ausführen einer Beispielabfrage

Um eine CloudWatch Logs Insights-Beispielabfrage auszuführen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.

2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Logs Insights aus.

Auf der Logs Insights-Seite enthält der Abfrage-Editor eine Standardabfrage in Logs Insights QL, die die 20 neuesten Protokollereignisse zurückgibt.

3. Wählen Sie in der Dropdownliste Select log group(s) (Protokollgruppe(n) auswählen) eine oder mehrere Protokollgruppen aus, die abgefragt werden sollen.

Wenn es sich um ein Überwachungskonto mit CloudWatch kontenübergreifender Observability handelt, können Sie Protokollgruppen sowohl in den Quellkonten als auch im Überwachungskonto auswählen. Mit einer einzigen Abfrage können Protokolle von verschiedenen Konten gleichzeitig abgefragt werden.

Sie können die Protokollgruppen nach dem Namen der Protokollgruppe, der Konto-ID oder der Kontobezeichnung filtern.

Wenn Sie eine Protokollgruppe in der Standard-Protokollklasse auswählen, erkennt CloudWatch Logs Insights automatisch Datenfelder in der Gruppe. Um diese erkannten Felder anzuzeigen, wählen Sie das Menü Fields (Felder) oben rechts auf der Seite aus.

 Note

Entdeckte Felder werden nur für Protokollgruppen der Standard-Protokollklasse unterstützt. Weitere Hinweise zu Protokollklassen finden Sie unter [Klassen protokollieren](#).

4. (Optional) Verwenden Sie die Zeitintervallauswahl, um einen Zeitraum auszuwählen, den Sie abfragen möchten.

Sie können zwischen Intervallen von 5 bis 30 Minuten, Intervallen von 1, 3 und 12 Stunden oder einem benutzerdefinierten Zeitrahmen wählen.

5. Klicken Sie auf Run (Ausführen), um die Ergebnisse anzuzeigen.

Für dieses Tutorial enthalten die Ergebnisse die 20 zuletzt hinzugefügten Protokollereignisse.

CloudWatch Logs zeigt ein Balkendiagramm der Protokollereignisse in der Protokollgruppe im Zeitverlauf an. Das Balkendiagramm zeigt nicht nur die Ereignisse in der Tabelle, sondern auch die Verteilung der Ereignisse in der Protokollgruppe, die der Abfrage und dem Zeitrahmen entsprechen.

6. Um alle Felder für ein zurückgegebenes Protokollereignis anzuzeigen, wählen Sie das dreieckige Dropdown-Symbol links neben dem nummerierten Ereignis.

Ändern der Beispielabfrage

In diesem Tutorial ändern Sie die Beispielabfrage, um die 50 neuesten Protokollereignisse anzuzeigen.

Wenn Sie das vorherige Tutorial noch nicht ausgeführt haben, tun Sie das jetzt. Dieses Tutorial beginnt dort, wo das vorherige Tutorial endet.

Note

In einigen Beispielabfragen, die mit CloudWatch Logs Insights bereitgestellt werden, `tail` werden Befehle `head` oder anstelle von verwendet `limit`. Diese Befehle sind veraltet und wurden durch `limit` ersetzt. Verwenden Sie in allen von Ihnen geschriebenen Abfragen `limit` anstelle von `head` oder `tail`.

Um die CloudWatch Logs Insights-Beispielabfrage zu ändern

1. Ändern Sie im Abfrage-Editor 20 in 50. Wählen Sie anschließend Run (Ausführen) aus.

Die Ergebnisse der neuen Abfrage werden angezeigt. Unter der Annahme, dass in der Protokollgruppe im Standardzeitraum genügend Daten vorhanden sind, werden nun 50 Protokollereignisse aufgelistet.

2. (Optional) Sie können Abfragen speichern, die Sie erstellt haben. Um diese Abfrage zu speichern, wählen Sie Speichern aus. Weitere Informationen finden Sie unter [Speichern Sie CloudWatch Logs Insights-Abfragen und führen Sie sie erneut aus](#).

Hinzufügen eines Filterbefehls zur Beispielabfrage

Dieses Tutorial zeigt, wie Sie eine umfangreichere Änderung der Abfrage im Abfrageeditor vornehmen können. In diesem Tutorial filtern Sie die Ergebnisse der vorherigen Abfrage basierend auf einem Feld in den abgerufenen Protokollereignissen.

Wenn Sie die vorherigen Tutorials noch nicht ausgeführt haben, tun Sie das jetzt. Dieses Tutorial beginnt dort, wo das vorherige Tutorial endet.

So fügen Sie der vorherigen Abfrage einen Filterbefehl hinzu

1. Entscheiden Sie sich für ein zu filterndes Feld. Wählen Sie rechts auf der Seite Felder aus, um die häufigsten Felder anzuzeigen, die CloudWatch Logs in den letzten 15 Minuten in den

Protokollereignissen in den ausgewählten Protokollgruppen erkannt hat, sowie den Prozentsatz dieser Protokollereignisse, in denen jedes Feld vorkommt.

Um die in einem bestimmten Protokollereignis enthaltenen Felder anzuzeigen, wählen Sie das Symbol links neben dieser Zeile aus.

Möglicherweise wird das Feld `awsRegion` im Protokollereignis angezeigt. Dies ist von den Ereignissen abhängig, die in Ihren Protokollen enthalten sind. Im Rest dieses Tutorials verwenden Sie `awsRegion` als Filterfeld. Sie können jedoch auch ein anderes Feld verwenden, wenn dieses Feld nicht verfügbar ist.

2. Positionieren Sie den Cursor im Feld des Abfrageeditors hinter 50, und drücken Sie die Eingabetaste.
3. Geben Sie in der neuen Zeile zunächst `|` (das Pipe-Zeichen) und ein Leerzeichen ein. Befehle in einer CloudWatch Logs Insights-Abfrage müssen durch einen senkrechten Strich getrennt werden.
4. Geben Sie **`filter awsRegion="us-east-1"`** ein.
5. Klicken Sie auf Ausführen.

Die Abfrage läuft erneut und zeigt nun die 50 neuesten Ergebnisse an, die dem neuen Filter entsprechen.

Wenn Sie nach einem anderen Feld gefiltert haben und ein Fehlerergebnis erhalten haben, müssen Sie möglicherweise den Feldnamen maskieren. Wenn der Feldname auch andere als alphanumerische Zeichen enthält, müssen Sie vor und nach dem Feldnamen Backtick-Zeichen (```) verwenden (z. B. **``error-code`="102"`**).

Sie müssen die Backtick-Zeichen für Feldnamen verwenden, die auch andere als alphanumerische Zeichen enthalten, nicht jedoch für Werte. Werte sind stets von Anführungszeichen (`"`) umschlossen.

Logs Insights QL umfasst leistungsstarke Abfragefunktionen, darunter mehrere Befehle und Unterstützung für reguläre Ausdrücke sowie mathematische und statistische Operationen. Weitere Informationen finden Sie unter [CloudWatch Abfragesyntax in Logs Insights](#).

Tutorial: Ausführen einer Abfrage mit einer Aggregationsfunktion

Sie können Aggregationsfunktionen im Befehl `stats` sowie als Argumente für andere Funktionen verwenden. In diesem Tutorial führen Sie einen Abfragebefehl aus, der die Anzahl

der Protokollereignisse zählt, die ein angegebenes Feld enthalten. Der Abfragebefehl gibt eine Gesamtzahl zurück, die nach dem Wert oder den Werten des angegebenen Felds gruppiert ist. Weitere Informationen zu Aggregationsfunktionen finden Sie unter [Unterstützte Operationen und Funktionen](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

Ausführen einer Abfrage mit einer Aggregationsfunktion

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Logs Insights aus.
3. Vergewissern Sie sich, dass die Registerkarte Logs Insights QL ausgewählt ist.
4. Wählen Sie in der Dropdownliste Select log group(s) (Protokollgruppe(n) auswählen) eine oder mehrere Protokollgruppen aus, die abgefragt werden sollen.

Wenn es sich um ein Überwachungskonto mit CloudWatch kontenübergreifender Observability handelt, können Sie Protokollgruppen sowohl in den Quellkonten als auch im Überwachungskonto auswählen. Mit einer einzigen Abfrage können Protokolle von verschiedenen Konten gleichzeitig abgefragt werden.

Sie können die Protokollgruppen nach dem Namen der Protokollgruppe, der Konto-ID oder der Kontobezeichnung filtern.

Wenn Sie eine Protokollgruppe auswählen, erkennt CloudWatch Logs Insights automatisch Datenfelder in der Protokollgruppe, wenn es sich um eine Protokollgruppe der Standardklasse handelt. Um diese erkannten Felder anzuzeigen, wählen Sie das Menü Fields (Felder) oben rechts auf der Seite aus.

5. Löschen Sie die Standardabfrage im Abfrage-Editor und geben Sie den folgenden Befehl ein:

```
stats count(*) by fieldName
```

6. *fieldName* Ersetzen Sie es durch ein erkanntes Feld aus dem Menü „Felder“.

Das Menü „Felder“ befindet sich oben rechts auf der Seite und zeigt alle erkannten Felder an, die CloudWatch Logs Insights in Ihrer Protokollgruppe erkennt.

7. Klicken Sie auf Run (Ausführen), um die Abfrageergebnisse anzuzeigen.

Die Abfrageergebnisse zeigen die Anzahl der Datensätze in Ihrer Protokollgruppe, die mit dem Abfragebefehl übereinstimmen, und die Gesamtzahl, die nach dem Wert oder den Werten des angegebenen Felds gruppiert ist.

Tutorial: Ausführen einer Abfrage, die eine nach Protokollfeldern gruppierte Visualisierung erzeugt

Wenn Sie eine Abfrage ausführen, die die zurückgegebenen Ergebnisse über die Funktion `stats` nach den Werten von einem oder mehreren Feldern in den Protokolleinträgen gruppiert, können Sie die Ergebnisse als Balken-, Kreis-, Linien- oder gestapeltes Flächendiagramm aufrufen. Auf diese Weise können Sie Trends in Ihren Protokollen effizienter visualisieren.

So führen Sie eine Abfrage zur Visualisierung durch

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Logs Insights aus.
3. Wählen Sie in der Dropdownliste Select log group(s) (Protokollgruppe(n) auswählen) eine oder mehrere Protokollgruppen aus, die abgefragt werden sollen.

Wenn es sich um ein Überwachungskonto mit CloudWatch kontenübergreifender Observability handelt, können Sie Protokollgruppen sowohl in den Quellkonten als auch im Überwachungskonto auswählen. Mit einer einzigen Abfrage können Protokolle von verschiedenen Konten gleichzeitig abgefragt werden.

Sie können die Protokollgruppen nach dem Namen der Protokollgruppe, der Konto-ID oder der Kontobezeichnung filtern.

4. Löschen Sie im Abfrageeditor den aktuellen Inhalt, geben Sie dann die folgende `stats`-Funktion ein und klicken Sie auf Abfrage ausführen.

```
stats count(*) by @logStream  
| limit 100
```

Die Ergebnisse zeigen für jeden Protokolldatenstrom die Anzahl der Protokollereignisse in der Protokollgruppe. Die Ergebnisse sind auf nur 100 Zeilen begrenzt.

5. Wählen Sie die Registerkarte Visualization (Visualisierung) aus.
6. Wählen Sie den Pfeil neben Line (Zeile) aus. Wählen Sie anschließend Bar (Balken) aus.

Das Balkendiagramm wird angezeigt, wobei jeder Protokolldatenstrom in der Protokollgruppe von einem Balken dargestellt wird.

Tutorial: Ausführen einer Abfrage, die eine Visualisierung von Zeitreihen erzeugt

Wenn Sie eine Abfrage ausführen, die die zurückgegebenen Ergebnisse über die Funktion `bin()` nach einem Zeitraum gruppiert, können Sie die Ergebnisse als Linien-, Balken- oder gestapeltes Flächendiagramm anzeigen. Auf diese Weise können Sie Trends in Protokollereignissen im Laufe der Zeit effizienter visualisieren.

So führen Sie eine Abfrage zur Visualisierung durch

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Logs Insights aus.
3. Vergewissern Sie sich, dass die Registerkarte Logs Insights QL ausgewählt ist.
4. Wählen Sie in der Dropdownliste Select log group(s) (Protokollgruppe(n) auswählen) eine oder mehrere Protokollgruppen aus, die abgefragt werden sollen.

Wenn es sich um ein Überwachungskonto mit CloudWatch kontenübergreifender Observability handelt, können Sie Protokollgruppen sowohl in den Quellkonten als auch im Überwachungskonto auswählen. Mit einer einzigen Abfrage können Protokolle von verschiedenen Konten gleichzeitig abgefragt werden.

Sie können die Protokollgruppen nach dem Namen der Protokollgruppe, der Konto-ID oder der Kontobezeichnung filtern.

5. Löschen Sie im Abfrageeditor den aktuellen Inhalt, geben Sie dann die folgende `stats`-Funktion ein und klicken Sie auf Abfrage ausführen.

```
stats count(*) by bin(30s)
```

Die Ergebnisse zeigen die Anzahl der Protokollereignisse in der Protokollgruppe, die von CloudWatch Logs für jeden Zeitraum von 30 Sekunden empfangen wurden.

6. Wählen Sie die Registerkarte Visualization (Visualisierung) aus.

Die Ergebnisse werden als Liniendiagramm dargestellt. Um zu einem Balken-, Kreis- oder gestapelten Flächendiagramm zu wechseln, wählen Sie oben Links im Diagramm auf den Pfeil neben Line (Linie).

Beispielabfragen

Dieser Abschnitt enthält eine Liste allgemeiner und nützlicher Abfragebefehle, die Sie in der [CloudWatchKonsole](#) ausführen können. Informationen zum Ausführen eines Abfragebefehls finden Sie unter [Tutorial: Eine Beispielabfrage ausführen und ändern](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

Weitere Informationen zur Abfragesyntax finden Sie unter [CloudWatch Abfragesyntax in Logs Insights](#).

Themen

- [Allgemeine Abfragen](#)
- [Abfragen für Lambda-Protokolle](#)
- [Abfragen für Flussprotokolle von Amazon-VPC](#)
- [Abfragen für Route-53-Protokolle](#)
- [Abfragen für CloudTrail Protokolle](#)
- [Abfragen für Amazon API Gateway](#)
- [Abfragen für NAT-Gateway](#)
- [Abfragen für Apache-Serverprotokolle](#)
- [Anfragen für Amazon EventBridge](#)
- [Beispiele des parse-Befehls](#)

Allgemeine Abfragen

Findet die 25 zuletzt hinzugefügten Protokollereignisse.

```
fields @timestamp, @message | sort @timestamp desc | limit 25
```

Ruft eine Liste der Anzahl der Ausnahmen pro Stunde ab.

```
filter @message like /Exception/  
  | stats count(*) as exceptionCount by bin(1h)  
  | sort exceptionCount desc
```

Ruft eine Liste von Protokollereignissen ab, die keine Ausnahmen sind.

```
fields @message | filter @message not like /Exception/
```

Ruft das letzte Protokollereignis für jeden eindeutigen Wert des Felds **server** ab.

```
fields @timestamp, server, severity, message  
| sort @timestamp asc  
| dedup server
```

Ruft das letzte Protokollereignis für jeden eindeutigen Wert des Felds **server** für jeden **severity**-Typ ab.

```
fields @timestamp, server, severity, message  
| sort @timestamp desc  
| dedup server, severity
```

Abfragen für Lambda-Protokolle

Ermittelt die Menge des zu viel bereitgestellten Speichers.

```
filter @type = "REPORT"  
| stats max(@memorySize / 1000 / 1000) as provisionedMemoryMB,  
    min(@maxMemoryUsed / 1000 / 1000) as smallestMemoryRequestMB,  
    avg(@maxMemoryUsed / 1000 / 1000) as avgMemoryUsedMB,  
    max(@maxMemoryUsed / 1000 / 1000) as maxMemoryUsedMB,  
    provisionedMemoryMB - maxMemoryUsedMB as overProvisionedMB
```

Erstellt einen Latenzbericht.

```
filter @type = "REPORT" |  
    stats avg(@duration), max(@duration), min(@duration) by bin(5m)
```

Sucht nach langsamen Funktionsaufrufen und beseitigt doppelte Anfragen, die durch Wiederholungen oder clientseitigen Code entstehen können. In dieser Abfrage ist **@duration** in Millisekunden.

```
fields @timestamp, @requestId, @message, @logStream  
| filter @type = "REPORT" and @duration > 1000  
| sort @timestamp desc
```

```
| dedup @requestId  
| limit 20
```

Abfragen für Flussprotokolle von Amazon-VPC

Findet die Top 15 Paketübertragungen zwischen den Hosts:

```
stats sum(packets) as packetsTransferred by srcAddr, dstAddr  
| sort packetsTransferred desc  
| limit 15
```

Findet die Top 15 Byte-Übertragungen für Hosts in einem bestimmten Subnetz.

```
filter isIpv4InSubnet(srcAddr, "192.0.2.0/24")  
| stats sum(bytes) as bytesTransferred by dstAddr  
| sort bytesTransferred desc  
| limit 15
```

Findet die IP-Adressen, die UDP als Datenübertragungsprotokoll verwenden.

```
filter protocol=17 | stats count(*) by srcAddr
```

Findet die IP-Adressen, bei denen während des Erfassungsfensters Flussdatensätze übersprungen wurden.

```
filter logStatus="SKIPDATA"  
| stats count(*) by bin(1h) as t  
| sort t
```

Sucht einen einzelnen Datensatz für jede Verbindung, um Probleme mit der Netzwerkverbindung zu beheben.

```
fields @timestamp, srcAddr, dstAddr, srcPort, dstPort, protocol, bytes  
| filter logStream = 'vpc-flow-logs' and interfaceId = 'eni-0123456789abcdef0'  
| sort @timestamp desc  
| dedup srcAddr, dstAddr, srcPort, dstPort, protocol  
| limit 20
```

Abfragen für Route-53-Protokolle

Findet die Verteilung der Datensätze pro Stunde nach Abfragetyp.

```
stats count(*) by queryType, bin(1h)
```

Findet die 10 DNS-Resolver mit der höchsten Anzahl von Anforderungen.

```
stats count(*) as numRequests by resolverIp  
  | sort numRequests desc  
  | limit 10
```

Ermittelt die Anzahl der Datensätze nach Domain und Subdomain, bei denen der Server die DNS-Anforderung nicht abgeschlossen hat.

```
filter responseCode="SERVFAIL" | stats count(*) by queryName
```

Abfragen für CloudTrail Protokolle

Ermitteln Sie die Anzahl der Protokolleinträge für jeden Dienst, Ereignistyp und AWS Region.

```
stats count(*) by eventSource, eventName, awsRegion
```

Finden Sie die Amazon EC2 EC2-Hosts, die in einer bestimmten AWS Region gestartet oder gestoppt wurden.

```
filter (eventName="StartInstances" or eventName="StopInstances") and awsRegion="us-east-2"
```

Finden Sie die AWS Regionen, Benutzernamen und ARNs neu erstellter IAM-Benutzer.

```
filter eventName="CreateUser"  
  | fields awsRegion, requestParameters.userName, responseElements.user.arn
```

Ermittelt die Anzahl der Datensätze, bei denen beim Aufruf des API-**UpdateTrail** eine Ausnahme aufgetreten ist.

```
filter eventName="UpdateTrail" and ispresent(errorCode)
| stats count(*) by errorCode, errorMessage
```

Findet Protokolleinträge, in denen TLS 1.0 oder 1.1 verwendet wurde.

```
filter tlsDetails.tlsVersion in [ "TLSv1", "TLSv1.1" ]
| stats count(*) as numOutdatedTlsCalls by userIdentity.accountId, recipientAccountId,
eventSource, eventName, awsRegion, tlsDetails.tlsVersion, tlsDetails.cipherSuite,
userAgent
| sort eventSource, eventName, awsRegion, tlsDetails.tlsVersion
```

Ermittelt die Anzahl der Anrufe pro Service, die die TLS-Versionen 1.0 oder 1.1 verwendet haben.

```
filter tlsDetails.tlsVersion in [ "TLSv1", "TLSv1.1" ]
| stats count(*) as numOutdatedTlsCalls by eventSource
| sort numOutdatedTlsCalls desc
```

Abfragen für Amazon API Gateway

Findet die letzten 10 4XX-Fehler.

```
fields @timestamp, status, ip, path, httpMethod
| filter status>=400 and status<=499
| sort @timestamp desc
| limit 10
```

Identifizieren Sie die 10 Amazon API Gateway Anfragen mit der längsten Laufzeit in Ihrer Amazon API Gateway Zugriffs-Log-Gruppe

```
fields @timestamp, status, ip, path, httpMethod, responseLatency
| sort responseLatency desc
| limit 10
```

Gibt die Liste der beliebtesten API-Pfade in Ihrer Amazon API Gateway -Zugriffsprotokollgruppe zurück.

```
stats count(*) as requestCount by path
```

```
| sort requestCount desc
| limit 10
```

Erstellt einen Integrationslatenzbericht für Ihre Amazon API Gateway -Zugriffsprotokollgruppe.

```
filter status=200
| stats avg(integrationLatency), max(integrationLatency),
min(integrationLatency) by bin(1m)
```

Abfragen für NAT-Gateway

Wenn Sie feststellen, dass Ihre AWS Rechnung höhere als normale Kosten enthält, können Sie CloudWatch Logs Insights verwenden, um die wichtigsten Mitwirkenden zu finden. Weitere Informationen zu den folgenden Abfragebefehlen finden Sie unter [Wie finde ich die Hauptverursacher des Datenverkehrs über das NAT-Gateway in meiner VPC?](#) auf der AWS Premium-Support-Seite.

Note

Ersetzen Sie in den folgenden Abfragebefehlen „x.x.x.x“ durch die private IP Ihres NAT-Gateways und ersetzen Sie „y.y.“ durch die ersten beiden Oktette Ihres VPC-CIDR-Bereichs.

Findet die Instances, die den meisten Datenverkehr über Ihr NAT-Gateway senden.

```
filter (dstAddr like 'x.x.x.x' and srcAddr like 'y.y.')
| stats sum(bytes) as bytesTransferred by srcAddr, dstAddr
| sort bytesTransferred desc
| limit 10
```

Bestimmt den Datenverkehr, der bei den Instances in Ihren NAT-Gateways ein- bzw. ausgeht.

```
filter (dstAddr like 'x.x.x.x' and srcAddr like 'y.y.') or (srcAddr like 'xxx.xx.xx.xx'
and dstAddr like 'y.y.')
| stats sum(bytes) as bytesTransferred by srcAddr, dstAddr
| sort bytesTransferred desc
| limit 10
```

Bestimmt die Internetziele, mit denen die Instances in Ihrer VPC am häufigsten für Uploads und Downloads kommunizieren.

For uploads (Für Uploads)

```
filter (srcAddr like 'x.x.x.x' and dstAddr not like 'y.y.')
| stats sum(bytes) as bytesTransferred by srcAddr, dstAddr
| sort bytesTransferred desc
| limit 10
```

Für Downloads

```
filter (dstAddr like 'x.x.x.x' and srcAddr not like 'y.y.')
| stats sum(bytes) as bytesTransferred by srcAddr, dstAddr
| sort bytesTransferred desc
| limit 10
```

Abfragen für Apache-Serverprotokolle

Sie können CloudWatch Logs Insights verwenden, um Apache-Serverprotokolle abzufragen. Weitere Informationen zu den folgenden Abfragen finden Sie unter [Simplifying Apache Server Logs with CloudWatch Logs Insights](#) im AWS Cloud Operations & Migrations Blog.

Findet die relevantesten Felder, damit Sie Ihre Zugriffsprotokolle und den Datenverkehr im Pfad / admin Ihrer Anwendung überprüfen können.

```
fields @timestamp, remoteIP, request, status, filename| sort @timestamp desc
| filter filename="/var/www/html/admin"
| limit 20
```

Ermittelt die Anzahl der eindeutigen GET-Anforderungen, die auf Ihre Hauptseite mit dem Statuscode „200“ (Erfolg) zugegriffen haben.

```
fields @timestamp, remoteIP, method, status
| filter status="200" and referrer= http://34.250.27.141/ and method= "GET"
| stats count_distinct(remoteIP) as UniqueVisits
| limit 10
```

Gibt an, wie oft Ihr Apache-Service neu gestartet wurde.

```
fields @timestamp, function, process, message
| filter message like "resuming normal operations"
```

```
| sort @timestamp desc
| limit 20
```

Anfragen für Amazon EventBridge

Ruft die Anzahl der EventBridge Ereignisse ab, gruppiert nach Art der Ereignisdetails

```
fields @timestamp, @message
| stats count(*) as numberOfEvents by `detail-type`
| sort numberOfEvents desc
```

Beispiele des parse-Befehls

Verwenden Sie einen globalen Ausdruck zum Extrahieren der Felder **@user**, **@method** und **@latency** aus dem Protokollfeld **@message** und zur Rückgabe der durchschnittlichen Latenz für jede eindeutige Kombination aus **@method** und **@user**.

```
parse @message "user=*, method:*, latency := *" as @user,
    @method, @latency | stats avg(@latency) by @method,
    @user
```

Verwenden Sie einen regulären Ausdruck zum Extrahieren der Felder **@user2**, **@method2** und **@latency2** aus dem Protokollfeld **@message** und zur Rückgabe der durchschnittlichen Latenz für jede eindeutige Kombination aus **@method2** und **@user2**.

```
parse @message /user=(?<user2>.*?), method:(?<method2>.*?),
    latency := (?<latency2>.*?)/ | stats avg(latency2) by @method2,
    @user2
```

Extrahiert die Felder **loggingTime**, **loggingType** und **loggingMessage**, filtert nach Protokollereignissen, die die Zeichenfolgen **ERROR** oder **INFO** enthalten, und zeigt dann nur die Felder **loggingMessage** und **loggingType** für Ereignisse an, die die Zeichenfolge **ERROR** enthalten.

```
FIELDS @message
| PARSE @message "*" [*] "*" as loggingTime, loggingType, loggingMessage
| FILTER loggingType IN ["ERROR", "INFO"]
| DISPLAY loggingMessage, loggingType = "ERROR" as isError
```


Vergleiche (Diff) mit früheren Zeitbereichen

Sie können CloudWatch Logs Insights mit Logs Insights QL verwenden, um Änderungen Ihrer Protokollereignisse im Laufe der Zeit zu vergleichen. Sie können die in einem kürzlichen Zeitraum aufgenommenen Protokollereignisse mit den Protokollen aus dem unmittelbar vorangegangenen Zeitraum vergleichen. Alternativ können Sie Vergleiche mit ähnlichen vergangenen Zeiträumen durchführen. Auf diese Weise können Sie herausfinden, ob ein Fehler in Ihren Protokollen erst vor Kurzem aufgetreten ist oder bereits aufgetreten ist, und es kann Ihnen auch helfen, andere Trends zu finden.

Vergleichsabfragen geben nur Muster in den Ergebnissen zurück, keine Rohprotokollereignisse. Anhand der zurückgegebenen Muster können Sie schnell die Trends und Änderungen der Protokollereignisse im Laufe der Zeit erkennen. Nachdem Sie eine Vergleichsabfrage ausgeführt haben und die Musterergebnisse vorliegen, können Sie sich beispielhafte Rohprotokollereignisse für die Muster ansehen, an denen Sie interessiert sind. Weitere Informationen zu Protokollmustern finden Sie unter [Musteranalyse](#).

Wenn Sie eine Vergleichsabfrage ausführen, wird Ihre Abfrage anhand von zwei verschiedenen Zeiträumen analysiert: dem ursprünglichen Abfragezeitraum, den Sie ausgewählt haben, und dem Vergleichszeitraum. Der Vergleichszeitraum hat immer die gleiche Länge wie Ihr ursprünglicher Abfragezeitraum. Die Standardzeitintervalle für die Vergleiche sind die folgenden.

- Vorheriger Zeitraum — Vergleicht den Zeitraum unmittelbar vor dem Abfragezeitraum.
- Vorheriger Tag — Vergleicht den Zeitraum einen Tag vor Ihrem Abfragezeitraum.
- Vorige Woche — Vergleicht den Zeitraum eine Woche vor Ihrem Abfragezeitraum.
- Vorheriger Monat — Vergleicht den Zeitraum einen Monat vor Ihrem Abfragezeitraum.

Note

Für Abfragen, die Vergleiche verwenden, fallen Gebühren an, die denen der Ausführung einer einzelnen CloudWatch Logs Insights-Abfrage über den gesamten Zeitraum ähneln. Weitere Informationen finden Sie unter [Amazon CloudWatch – Preise](#).

Um eine Vergleichsabfrage auszuführen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.

2. Wählen Sie im Navigationsbereich Protokolle und dann Logs Insights aus.

Im Abfragefeld wird eine Standardabfrage angezeigt.

3. Vergewissern Sie sich, dass die Registerkarte Logs Insights QL ausgewählt ist.
4. Behalten Sie die Standardabfrage bei oder geben Sie eine andere Abfrage ein.
5. Wählen Sie in der Dropdownliste Protokollgruppe (n) auswählen eine oder mehrere Protokollgruppen für die Abfrage aus.
6. (Optional) Verwenden Sie die Zeitintervallauswahl, um einen Zeitraum auszuwählen, den Sie abfragen möchten. Die Standardabfrage bezieht sich auf die Protokolldaten der letzten Stunde.
7. Wählen Sie in der Zeitbereichsauswahl die Option Vergleichen aus. Wählen Sie dann den vorherigen Zeitraum aus, mit dem Sie die Originalprotokolle vergleichen möchten, und klicken Sie auf Anwenden.
8. Wählen Sie Abfrage ausführen.

Damit die Abfrage die Daten aus dem Vergleichszeitraum abrufen, wird der `diff` Befehl an Ihre Abfrage angehängt.

9. Wählen Sie die Registerkarte Muster, um die Ergebnisse zu sehen.

In der Tabelle werden die folgenden Informationen angezeigt:

- Jedes Muster, wobei variable Teile des Musters durch das dynamische Tokensymbol ersetzt werden `<string-number>`. Das `string` ist eine Beschreibung des Datentyps, für den das Token steht. Das `number` zeigt, an welcher Stelle des Musters dieses Token im Vergleich zu den anderen dynamischen Tokens erscheint. Weitere Informationen finden Sie unter [Musteranalyse](#).
- Die Anzahl der Ereignisse ist die Anzahl der Protokollereignisse mit diesem Muster im ursprünglichen, aktuelleren Zeitraum.
- Die Anzahl der Differenzereignisse ist die Differenz zwischen der Anzahl übereinstimmender Protokollereignisse im aktuellen Zeitraum und der Anzahl der Vergleichszeiträume. Ein positiver Unterschied bedeutet, dass es im aktuellen Zeitraum mehr solcher Ereignisse gibt.
- Die Beschreibung des Unterschieds fasst kurz die Änderung dieses Musters zwischen dem aktuellen Zeitraum und dem Vergleichszeitraum zusammen.
- Der Schweregradtyp ist der wahrscheinliche Schweregrad der Protokollereignisse mit diesem Muster, basierend auf Wörtern, die in den Protokollereignissen vorkommen FATAL, wie ERROR, und. WARN

10. Um eines der Muster in der Liste genauer zu untersuchen, wählen Sie das Symbol in der Spalte Inspizieren für eines der Muster aus.

Das Fenster „Muster prüfen“ wird geöffnet und enthält Folgendes:

- Das Muster. Wählen Sie ein Token innerhalb des Musters aus, um die Werte dieses Tokens zu analysieren.
- Ein Histogramm, das die Anzahl der Vorkommen des Musters im abgefragten Zeitraum zeigt. Dies kann Ihnen helfen, interessante Trends zu identifizieren, z. B. eine plötzliche Zunahme des Auftretens eines Musters.
- Auf der Registerkarte Protokollbeispiele werden einige der Protokollereignisse angezeigt, die dem ausgewählten Muster entsprechen.
- Auf der Registerkarte Token-Werte werden die Werte des ausgewählten dynamischen Tokens angezeigt, sofern Sie eines ausgewählt haben.

Note

Für jedes Token werden maximal 10 Token-Werte erfasst. Die Anzahl der Tokens ist möglicherweise nicht genau. CloudWatch Logs verwendet einen probabilistischen Zähler, um die Tokenanzahl zu generieren, nicht den absoluten Wert.

- Auf der Registerkarte Verwandte Muster werden andere Muster angezeigt, die häufig fast zur gleichen Zeit wie das Muster aufgetreten sind, das Sie untersuchen. Wenn beispielsweise ein Muster für eine ERROR Nachricht normalerweise von einem anderen Protokollereignis begleitet wurde, das als INFO mit zusätzlichen Details gekennzeichnet war, wird dieses Muster hier angezeigt.

Visualisieren von Protokolldaten in Diagrammen

Sie können Visualisierungen wie Balkendiagramme, Liniendiagramme und gestapelte Flächendiagramme verwenden, um Muster in Ihren Protokolldaten effizienter zu identifizieren. CloudWatch Logs Insights generiert Visualisierungen für Abfragen, die die `stats` Funktion und eine oder mehrere Aggregationsfunktionen verwenden. Weitere Informationen finden Sie unter [stats](#).

OpenSearch Piped Processing Language (PPL)

Dieser Abschnitt enthält eine grundlegende Einführung in das Abfragen von CloudWatch Logs mit PPL. OpenSearch Mit PPL können Sie Daten mithilfe von in der Pipeline hinterlegten Befehlen

abrufen, abfragen und analysieren, was das Verständnis und die Erstellung komplexer Abfragen erleichtert. Die Syntax basiert auf Unix-Pipes und ermöglicht die Verkettung von Befehlen zur Transformation und Verarbeitung von Daten. Mit PPL können Sie Daten filtern und aggregieren und eine Vielzahl von mathematischen, Zeichenketten-, Datums-, Bedingungs- und anderen Funktionen für die Analyse verwenden.

Die Aufnahme `SOURCE` in eine PPL-Abfrage ist eine nützliche Methode, um die Protokollgruppen, Feldindizes und Datenquellen anzugeben, die in eine Abfrage aufgenommen werden sollen, wenn Sie die AWS CLI OR-API zum Erstellen einer Abfrage verwenden. Der `SOURCE` Befehl wird nur in der AWS CLI AND-API unterstützt, nicht in der CloudWatch Konsole. Wenn Sie die CloudWatch Konsole verwenden, um eine Abfrage zu starten, verwenden Sie die Konsolenschnittstelle, um die Protokollgruppen sowie den Namen und Typ der Datenquelle anzugeben.

Wird verwendet, `aws:fieldIndex` um nur indizierte Daten zurückzugeben, indem eine Abfrage erzwungen wird, nur Protokollgruppen zu scannen, die in einem Feld indexiert sind, das Sie in der Abfrage angeben. Die relevanten Protokollgruppen werden automatisch auf der Grundlage der im Befehl angegebenen Felder ausgewählt. `filterIndex` Dadurch wird das gescannte Volumen reduziert, indem Protokollgruppen übersprungen werden, die keine Protokollereignisse enthalten, die das in der Abfrage angegebene Feld enthalten, und nur Protokollgruppen gescannt werden, die dem in der Abfrage für diesen Feldindex angegebenen Wert entsprechen. Geben Sie `aws:fieldIndex` hier den Feldnamen zusammen mit dem Feldnamen und dem Feldwert im Quellbefehl an, um nur indizierte Daten abzufragen, die das angegebene Feld und den angegebenen Wert enthalten. Weitere Informationen finden Sie unter [Erstellen Sie Feldindizes, um die Abfrageleistung zu verbessern und das Scanvolumen zu reduzieren](#).

Sie können OpenSearch PPL für Abfragen von Protokollgruppen in der Standard-Protokollklasse verwenden.

Note

Informationen zu allen in CloudWatch Logs unterstützten OpenSearch PPL-Abfragebefehlen sowie ausführliche Informationen zu Syntax und Einschränkungen finden Sie unter [Unterstützte PPL-Befehle](#) im OpenSearch Service Developer Guide.

Informationen zu anderen Abfragesprachen, die Sie verwenden können, finden Sie unter [CloudWatch Logs Insights](#), [OpenSearch Service SQL](#) und [CloudWatch Metrics Insights](#)

Befehl oder Funktion	Beispielabfrage	Description
<code>fields</code>	<code>fields field1, field2</code>	Zeigt eine Reihe von Feldern an, die projiziert werden müssen.
<code>join</code>	<code>LEFT JOIN left=l, right=r on l.id = r.id `join_right_lg` fields l.field_1, r.field_2</code>	Verbindet zwei Datensätze miteinander.
<code>where</code>	<code>where field1="success" where field2 != "i-023fe0a90929d8822" fields field3, field4, field5,field6 head 1000</code>	Filtert die Daten auf der Grundlage der von Ihnen angegebenen Bedingungen.
<code>aws:field Index</code>	<code>source = [`aws:fieldIndex`="region", `region` = "us-west-2"] where status = 200 head 10</code>	Gibt nur indizierte Daten zurück, indem eine Abfrage erzwungen wird, nur Protokollgruppen zu scannen, die für ein Feld indiziert sind, das Sie in der Abfrage angeben.
<code>stats</code>	<code>stats count(), count(field1), min(field 1), max(field1), avg(field1) by field2 head 1000</code>	Führt Aggregationen und Berechnungen durch

Befehl oder Funktion	Beispielabfrage	Description
parse	<pre>parse field1 ".*/(?<field2>[^/]+\$)" where field2 = "requestId" fields field1, field2 head 1000</pre>	Extrahiert ein Muster mit regulären Ausdrücken (Regex) aus einer Zeichenfolge und zeigt das extrahierte Muster an. Das extrahierte Muster kann weiter verwendet werden, um neue Felder zu erstellen oder Daten zu filtern.
sort	<pre>stats count(), count(field1), min(field1) as field1Alias, max(`field1`), avg(`field1`) by field2 sort -field1Alias head 1000</pre>	Sortiert die angezeigten Ergebnisse nach einem Feldnamen. Verwenden Sie <code>sort -FieldName</code> , um in absteigender Reihenfolge zu sortieren.

Befehl oder Funktion	Beispielabfrage	Description
eval	<pre>eval field2 = field1 * 2 fields field1, field2 head 20</pre>	Ändert oder verarbeitet den Wert eines Felds und speichert ihn in einem anderen Feld. Dies ist nützlich, um eine Spalte mathematisch zu modifizieren, Zeichenkettenfunktionen auf eine Spalte anzuwenden oder Datumsfunktionen auf eine Spalte anzuwenden.
rename	<pre>rename field2 as field1 fields field1;</pre>	Benennt ein oder mehrere Felder im Suchergebnis um.
head	<pre>fields `@message` head 20</pre>	Beschränkt die angezeigten Abfrageergebnisse auf die ersten N Zeilen.
top	<pre>top 2 field1 by field2</pre>	Findet die häufigsten Werte für ein Feld.

Befehl oder Funktion	Beispielabfrage	Description
dedup	<code>dedup field1 fields field1, field2, field3</code>	Entfernt doppelte Einträge auf der Grundlage der von Ihnen angegebenen Felder.
rare	<code>rare field1 by field2</code>	Findet die seltensten Werte aller Felder in der Feldliste.
subquery	<code>where field_1 IN [search source=`subquery_lg` fields field_2] fields id, field_1</code>	Führt komplexe, verschachtelte Abfragen innerhalb Ihrer PPL-Anweisungen durch.
trendline	<code>trendline sma(2, field1) as field1Alias</code>	Berechnet die gleitenden Durchschnitte von Feldern.

Befehl oder Funktion	Beispielabfrage	Description
eventStats	eventstats sum(field1) by field2	Reichert Ihre Ereignisdaten mit berechneten zusammenfassenden Statistiken an. Es analysiert bestimmte Felder innerhalb Ihrer Ereignisse, berechnet verschiedene statistische Kennzahlen und fügt diese Ergebnisse dann als neue Felder an jedes ursprüngliche Ereignis an.
expand	eval tags_array_string = json_extract(`@message`, '\$.tags') eval tags_array = json_array(json_extract(tags_string, '\$[0]'), json_extract(tags_string, '\$[1]')) expand tags_array as color_tags	Zerlegt ein Feld, das mehrere Werte enthält, in separate Zeilen und erstellt für jeden Wert im angegebenen Feld eine neue Zeile.

Befehl oder Funktion	Beispielabfrage	Description
<code>fillnull</code>	<code>fields `@timestamp`, error_code, status_code fillnull using status_code = "UNKNOWN", error_code = "UNKNOWN"</code>	Füllt Nullfelder mit dem von Ihnen angegebenen Wert. Es kann in einem oder mehreren Feldern verwendet werden.
<code>flatten</code>	<code>eval metadata_struct = json_object('size', json_extract(metadata_string, '\$.size'), 'color', json_extract(metadata_string, '\$.color')) flatten metadata_struct as (meta_size, meta_color)</code>	Macht ein Feld flach. Das Feld muss von diesem Typ sein: <code>struct<?, ?></code> oder <code>array<struct<?, ?>></code>
<code>cidrmatch</code>	<code>where cidrmatch(ip, '2003:db8::/32') fields ip</code>	Prüft, ob die angegebene IP-Adresse innerhalb des angegebenen CIDR-Bereichs liegt.

Befehl oder Funktion	Beispielabfrage	Description
fieldsummary	<pre>where field1 != 200 fieldsummary includefields= field1 nulls=true</pre>	Berechnet grundlegende Statistiken für jedes Feld (Anzahl, eindeutige Anzahl, Min., Max, Durchschnitt, Standardwert und Mittelwert).
grok	<pre>grok email '.*@%{HOSTNAME:host}' fields email, host</pre>	Analysiert ein Textfeld mit einem Grok-Muster und hängt die Ergebnisse an das Suchergebnis an.

Befehl oder Funktion	Beispielabfrage	Description
Zeichenfolgenfunktionen	<pre>eval field1Len = LENGTH(field1) fields field1Len</pre>	Built-in Funktionen in PPL, die Zeichenketten- und Textdaten innerhalb von PPL-Abfragen manipulieren und transformieren können. Zum Beispiel das Konvertieren von Groß- und Kleinschreibung, das Kombinieren von Zeichenketten, das Extrahieren von Teilen und das Bereinigen von Text.

Befehl oder Funktion	Beispielabfrage	Description
Date-Time Funktionen	<pre>eval newDate = ADDDATE(DATE('2020-08-26'), 1) fields newDate</pre>	Built-in Funktionen zur Behandlung und Transformation von Datums- und Zeitstempeldaten in PPL-Abfragen. Zum Beispiel <code>date_add</code> , <code>date_format</code> , <code>datediff</code> , <code>date_sub</code> , <code>timestamp add</code> , <code>timestamp diff</code> , <code>current_timezone</code> , <code>utc_timestamp</code> und <code>current_date</code> .
Bedingungs-funktionen	<pre>eval field2 = isnull(field1) fields field2, field1, field3</pre>	Built-in Funktionen, die nach bestimmten Feldbedingungen suchen und Ausdrücke bedingt auswerten. Wenn beispielsweise <code>field1</code> null ist, wird <code>field2</code> zurückgegeben.

Befehl oder Funktion	Beispielabfrage	Description
Mathematische Funktionen	<pre>eval field2 = ACOS(field1) fields field1</pre>	Built-in Funktionen zur Durchführung mathematischer Berechnungen und Transformationen in PPL-Abfragen. Zum Beispiel abs (absoluter Wert), round (rundet Zahlen), sqrt (Quadratwurzel), pow (Potenzberechnung) und ceil (rundet auf die nächste Ganzzahl auf).
CryptoGraphic Funktionen	<pre>eval crypto = MD5(field) head 1000</pre>	Um den Hash eines bestimmten Feldes zu berechnen

Befehl oder Funktion	Beispielabfrage	Description
JSON-Funktionen	<pre>eval valid_json = json('[1,2,3,{"f1":1,"f2":[5,6]},4]') fields valid_json</pre>	Built-in Funktionen für den Umgang mit JSON, einschließlich Arrays, Extrahieren und Validieren. Zum Beispiel <code>json_object</code> , <code>json_array</code> , <code>to_json_string</code> , <code>json_array_length</code> , <code>json_extract</code> , <code>json_keys</code> und <code>json_valid</code> .

Gültigkeitsbereich der Abfrage

Das Einbeziehen von SOURCE in eine Abfrage ist eine nützliche Methode, um die Protokollgruppen anzugeben, die in eine Abfrage aufgenommen werden sollen, wenn Sie die AWS CLI OR-API zum Erstellen einer Abfrage verwenden. Der Befehl SOURCE wird nur in der AWS CLI AND-API unterstützt, nicht in der CloudWatch Konsole. Wenn Sie die CloudWatch Konsole verwenden, um eine Abfrage zu starten, verwenden Sie die Konsolenschnittstelle, um die Protokollgruppen sowie den Namen und Typ der Datenquelle anzugeben.

Der Quellbefehl von PPL unterstützt jetzt mehrere Möglichkeiten, sie anzugeben:

1. Gruppe protokollieren
2. Feldindizes — Neu
3. Datenquelle und Datentyp — Neu

Protokollgruppe

Die Quellenauswahl für die Protokollgruppe kann verwendet werden, wenn Kunden wissen, welche Protokollgruppe (n) genau durchsucht werden muss

```
source = [lg:`/aws/lambda/my-function`] | where status = 200 | head 10
```

Feldindizes

Die auf Feldindizes basierende Quellauswahl reduziert die Menge der abgefragten Daten, indem die Ergebnisse auf indizierte Daten beschränkt werden, wenn Ihre Filter auf indizierte Felder abzielen. Die relevanten Protokollgruppen werden automatisch auf der Grundlage der im Befehl angegebenen Felder ausgewählt. `filterIndex` Weitere Informationen zu Feldindizes und deren Erstellung finden Sie unter [Feldindizes erstellen, um die Abfrageleistung zu verbessern und das Scanvolumen zu reduzieren](#).

Wird verwendet, `aws:fieldIndex` um nur indizierte Daten zurückzugeben, indem eine Abfrage erzwungen wird, nur Protokollgruppen zu scannen, die für ein Feld indiziert sind, das Sie in der Abfrage angeben. Für diese Protokollgruppen, die in diesem Feld indexiert sind, wird die Abfrage weiter optimiert, indem die Protokollgruppen übersprungen werden, die keine Protokollereignisse enthalten, die das in der Abfrage für das indizierte Feld angegebene Feld enthalten. Das gescannte Volumen wird weiter reduziert, indem versucht wird, nur Protokollereignisse aus diesen Protokollgruppen zu scannen, die dem in der Abfrage für diesen Feldindex angegebenen Wert entsprechen. Weitere Informationen zu Feldindizes und deren Erstellung finden Sie unter [Feldindizes erstellen, um die Abfrageleistung zu verbessern und das Scanvolumen zu reduzieren](#).

`aws:fieldIndex` Wird in PPL verwendet, um anzugeben, welche Schlüssel-Wert-Paare als Indizes behandelt werden sollen. Die Syntax lautet wie folgt

```
source = [aws:fieldIndex`="region", `region` = "us-west-2"] | where status = 200 | head 10
```

wobei

1. `aws:fieldIndex`="region"` identifiziert die Region als Feldindex.
 - a. Hinweis: Statt `=` können Kunden `IN` verwenden, um mehrere Indizes anzugeben (Beispiel unten)
2. ``region`="us-west-2"` identifiziert die anzuwendende Filterbedingung
 - a. Hinweis: Statt `=` können Kunden `IN` verwenden, um mehrere Werte anzugeben (Beispiel unten)

Kunden können mehrere FieldIndexes wie folgt angeben

```
source = [`aws:fieldIndex` IN ("status", "region"), `status` = 200, `region` IN ("us-west-2", "us-east-1")] | head 10
```

Datenquelle und Typ

Die Datenquellen- und typbasierte Quellenauswahl kann verwendet werden, wenn Kunden wissen, welche Datenquellen genau abgefragt werden müssen. Diese Abfrage wird über eine oder mehrere Protokollgruppen ausgeführt, die die angegebene Datenquelle und den angegebenen Datentyp enthalten.

```
source = [ds:`data_source.type`] | where status = 200 | head 10
```

PPL wird für Datenquellenabfragen unterstützt

Um den Anwendungsfall für die Abfrage von Datenquellen in PPL zu unterstützen, können Sie die dynamische Quellenauswahlklausel verwenden. Mit dieser Syntax können Sie Datenquellen abfragen, indem Sie sie im Suchbefehl angeben. Sie können bis zu 10 Datenquellen angeben.

Syntax

```
source=[ds:`DataSource1.Type1`, ds:`DataSource2.Type2`, ...ds:`DataSourcen.Typen`]
```

Beispielabfrage

```
search source=[ds:`DataSource1.Type1`, ds:`DataSource2.Type2`] | fields field1, field2
```

Kombiniertes Beispiel

Kunden können alle Operatoren zur Quellenauswahl in beliebiger Reihenfolge angeben. Die Ergebnisse wären dann der Schnittpunkt aller angewandten Bedingungen.

Beispielsweise könnte `/aws/lambda/my-function-1` mehrere Datenquellen und -typen enthalten, einschließlich einer Vielzahl von Indizes. Wenn die folgende Abfrage ausgeführt wurde, enthielten die zurückgegebenen Ergebnisse nur Ereignisse von Quelle `DataSource1.Type1` und Typ, die den Kriterien „Status“ = 200 entsprechen.

```
search source=[
```

```
ds:`DataSource1.Type1`,  
lg:`/aws/lambda/my-function-1`,  
`aws:fieldIndex` IN ("status"), `status` = 200  
]
```

Einschränkungen

Die folgenden Einschränkungen gelten, wenn Sie OpenSearch PPL für Abfragen in Logs Insights verwenden. CloudWatch

- Sie können die Befehle `join` oder `subquery` nicht mit Datenquellenabfragen verwenden.

OpenSearch Strukturierte Abfragesprache (SQL)

Dieser Abschnitt enthält eine grundlegende Einführung in das Abfragen von CloudWatch Logs mit OpenSearch SQL. Es bietet eine vertraute Option, wenn Sie es gewohnt sind, mit relationalen Datenbanken zu arbeiten. OpenSearch SQL bietet einen Teil der SQL-Funktionalität und ist daher eine gute Wahl für Ad-hoc-Abfragen und Datenanalyseaufgaben. Mit OpenSearch SQL können Sie Befehle wie `SELECT`, `FROM`, `WHERE`, `GROUP BY`, `HAVING` und verschiedene andere SQL-Befehle und -Funktionen verwenden. Sie können JOINS über Protokollgruppen hinweg ausführen, Daten zwischen Protokollgruppen mithilfe von Unterabfragen korrelieren und die umfangreiche Palette von JSON-, mathematischen, Zeichenketten-, bedingten und anderen SQL-Funktionen verwenden, um leistungsstarke Analysen von Protokoll- und Sicherheitsdaten durchzuführen.

Wird verwendet, `filterIndex` um nur indizierte Daten zurückzugeben, indem eine Abfrage erzwungen wird, nur Protokollgruppen zu scannen, die für ein Feld indiziert sind, das Sie in der Abfrage angeben. Reduzieren Sie das gescannte Volumen, indem Sie Protokollgruppen überspringen, die keine Protokollereignisse enthalten, die das in der Abfrage angegebene Feld enthalten, und nur Protokollgruppen scannen, die dem in der Abfrage angegebenen Wert für diesen Feldindex entsprechen. Geben Sie `filterIndex` hier den Feldnamen zusammen mit dem Feldnamen und dem Feldwert an, um nur indizierte Daten abzufragen, die das angegebene Feld und den angegebenen Wert enthalten.

Sie können OpenSearch SQL für Abfragen von Protokollgruppen in der Standard-Protokollklasse verwenden. SQL unterstützt auch Abfragen mithilfe des Datenquellennamens und des Datenquellentyps.

Note

In der folgenden Tabelle sind die in CloudWatch Logs unterstützten SQL-Befehle und -Funktionen aufgeführt. Informationen zu allen OpenSearch SQL-Befehlen einschließlich der Syntax finden Sie unter [Unterstützte SQL-Befehle](#) im OpenSearch Service Developer Guide. Informationen zu anderen Abfragesprachen, die Sie verwenden können, finden Sie unter [CloudWatch Logs Insights](#), [OpenSearch Service PPL](#) und [CloudWatch Metrics Insights](#).

Unterstützte SQL-Befehle

Note

Ersetzen Sie in der Beispielabfragespalte nach `<logGroup>` Bedarf, je nachdem, welche Datenquelle Sie abfragen.

Befehl oder Funktion	Beispielabfrage	Description
SELECT	SELECT `@message`, Operation FROM `LogGroupA`	Zeigt projizierte Werte an.
FROM	SELECT `@message`, Operation FROM `LogGroupA`	Built-in Klausel, die die Quelltablelle (n) oder Ansicht (en) angibt, aus denen Daten abgerufen werden sollen, und unterstützt verschiedene Arten von Verknüpfungen und Unterabfragen.

Befehl oder Funktion	Beispielabfrage	Description
WHERE	<code>SELECT * FROM `LogGroupA` WHERE Operation = 'x'</code>	Filtert Protokollereignisse auf der Grundlage der angegebenen Feldkriterien.
filterIndex	<code>SELECT * FROM `filterIndex('region' = 'us-east-1')` WHERE status = 200 LIMIT 10;</code>	Gibt nur indizierte Daten zurück, indem eine Abfrage erzwungen wird, nur Protokollgruppen zu scannen, die für ein Feld indiziert sind, das Sie in der Abfrage angeben.
GROUP BY	<code>SELECT `@logStream`, COUNT(*) as log_count FROM `LogGroupA` GROUP BY `@logStream`</code>	Gruppiert Ereignisse nach Kategorien und ermittelt anhand von Statistiken den Durchschnitt.
HAVING	<code>SELECT `@logStream`, COUNT(*) as log_count FROM `LogGroupA` GROUP BY `@logStream` HAVING log_count > 100</code>	Filtert die Ergebnisse anhand der Gruppierungsbedingungen.

Befehl oder Funktion	Beispielabfrage	Description
ORDER BY	<code>SELECT * FROM `LogGroupA` ORDER BY `@timestamp` DESC</code>	Sortiert die Ergebnisse auf der Grundlage der Felder in der ORDER BY-Klausel. Sie können entweder in absteigender oder aufsteigender Reihenfolge sortieren.
JOIN	<code>SELECT A.`@message`, B.`@timestamp` FROM `LogGroupA` as A INNER JOIN `LogGroupB` as B ON A.`requestId` = B.`requestId`</code>	Verbindet die Ergebnisse für zwei Tabellen auf der Grundlage gemeinsamer Felder. Inner JOIN oder Left Outer Join müssen angegeben werden
LIMIT	<code>Select * from `LogGroupA` limit 10</code>	Beschränkt die angezeigten Abfrageergebnisse auf die ersten N Zeilen.

Befehl oder Funktion	Beispielabfrage	Description
Zeichenfolgenfunktionen	<code>SELECT upper(Operation) , lower(Operation), Operation FROM `LogGroupA`</code>	Built-in Funktionen in SQL, die Zeichenketten- und Textdaten in SQL-Abfragen manipulieren und transformieren können. Zum Beispiel das Konvertieren von Groß- und Kleinschreibung, das Kombinieren von Zeichenketten, das Extrahieren von Teilen und das Bereinigen von Text.

Befehl oder Funktion	Beispielabfrage	Description
Datumsfunktionen	<pre>SELECT current_date() as today, date_add(current_date(), 30) as thirty_days_later, last_day(current_date()) as month_end FROM `LogGroupA`</pre>	Built-in Funktionen für die Verarbeitung und Transformation von Datums- und Zeitstempeldaten in SQL-Abfragen. Zum Beispiel <code>date_add</code> , <code>date_format</code> , <code>datediff</code> und <code>current_date</code> .
Konditionale Funktionen	<pre>SELECT Operation, IF(Error > 0, 'High', 'Low') as error_category FROM `LogGroupA`;</pre>	Built-in Funktionen, die Aktionen auf der Grundlage bestimmter Bedingungen ausführen oder Ausdrücke bedingt auswerten. Zum Beispiel <code>CASE</code> und <code>IF</code> .

Befehl oder Funktion	Beispielabfrage	Description
Aggregationsfunktionen	<pre>SELECT AVG(bytes) as bytesWritten FROM `LogGroupA`</pre>	Built-in Funktionen, die Berechnungen für mehrere Zeilen durchführen, um einen einzelnen zusammengefassten Wert zu erzeugen. Zum Beispiel SUM, COUNT, AVG, MAX und MIN.
JSON-Funktionen	<pre>SELECT get_json_object(json_column, '\$.name') as name FROM `LogGroupA`</pre>	Built-in Funktionen zum Parsen, Extrahieren, Ändern und Abfragen von JSON-formatte Daten innerhalb von SQL-Abfragen (z. B. from_json, to_json, get_json_object, json_tuple), die die Manipulation von JSON-Strukturen in Datensätzen ermöglichen.

Befehl oder Funktion	Beispielabfrage	Description
Array-Funktionen	<pre>SELECT scores, size(scores) as length, array_contains(scores, 90) as has_90 FROM `LogGroupA`;</pre>	Built-in Funktionen für die Arbeit mit Spalten vom Typ Array in SQL-Abfragen, die Operationen wie den Zugriff auf, die Änderung und die Analyse von Array-Daten ermöglichen (z. B. size, explode, array_contains).

Befehl oder Funktion	Beispielabfrage	Description
Fensterfunktionen	<pre>SELECT field1, field2, RANK() OVER (ORDER BY field2 DESC) as field2Rank FROM `LogGroupA`;</pre>	Built-in Funktionen, die Berechnungen für eine bestimmte Gruppe von Zeilen durchführen, die sich auf die aktuelle Zeile (Fenster) beziehen, wodurch Operationen wie Rangfolge, laufende Summen und gleitende Durchschnitt ermöglicht werden. Zum Beispiel ROW_NUMBER, RANK, LAG und LEAD

Befehl oder Funktion	Beispielabfrage	Description
Konvertierungs-Funktionen	<pre>SELECT CAST('123' AS INT) as converted _number, CAST(123 AS STRING) as converted _string FROM `LogGroupA`</pre>	Built-in Funktionen zum Konvertieren von Daten von einem Typ in einen anderen innerhalb von SQL-Abfragen, wodurch Datentyptransformationen und Formatkonvertierungen ermöglicht werden. Zum Beispiel CAST, TO_DATE, TO_TIMESTAMP und BINARY.

Befehl oder Funktion	Beispielabfrage	Description
Prädikatsfunktionen	<pre>SELECT scores, size(scores) as length, array_contains(scores, 90) as has_90 FROM `LogGroupA`;</pre>	Built-in Funktionen, die Bedingungen auswerten und boolesche Werte (true/false) auf der Grundlage bestimmte r Kriterien oder Muster zurückgeben. Zum Beispiel IN, LIKE, BETWEEN, IS NULL und EXISTS.
Wählen Sie mehrere Protokollgruppen aus	<pre>SELECT lg1.field1, lg1.field2 from `logGroups( logGroupIdentifizier: ['LogGroup1', 'LogGroup2'])` as lg1 where lg1.field3= "Success"</pre>	Ermöglicht es Ihnen, mehrere Protokollgruppen in einer SELECT-Anweisung anzugeben
Wählen Sie mehrere Datenquellen aus	<pre>SELECT ds1.field1, ds1.field2 from `dataSource(['DataSource1', 'DataSou rce2'])` as ds1 where ds1.field3= "Success"</pre>	Ermöglicht die Angabe mehrerer Datenquellen in einer SELECT-Anweisung

Unterstütztes SQL für Abfragen mit mehreren Protokollgruppen

Um den Anwendungsfall für die Abfrage mehrerer Protokollgruppen in SQL zu unterstützen, können Sie den Befehl verwenden. `logGroups` Mit dieser Syntax können Sie mehrere Protokollgruppen abfragen, indem Sie sie im Befehl `FROM` angeben.

Syntax:

```
`logGroups(  
    logGroupIdentifier: ['LogGroup1', 'LogGroup2', ... 'LogGroupn']  
)
```

In dieser Syntax können Sie bis zu 50 Protokollgruppen im `logGroupIdentifier` Parameter angeben. Um in einem Monitoring-Konto auf Protokollgruppen zu verweisen, verwenden Sie ARNs anstelle von LogGroup Namen.

Beispielabfrage:

```
SELECT LG1.Column1, LG1.Column2 from `logGroups(  
    logGroupIdentifier: ['LogGroup1', 'LogGroup2']  
)` as LG1 WHERE LG1.Column1 = 'ABC'
```

Die folgende Syntax mit mehreren Protokollgruppen nach der `FROM` Anweisung wird bei der Abfrage von CloudWatch Logs NICHT unterstützt.

```
SELECT Column1, Column2 FROM 'LogGroup1', 'LogGroup2', ... 'LogGroupn'  
WHERE Column1 = 'ABC'
```

Unterstütztes SQL für Datenquellenabfragen

Um den Anwendungsfall für die Abfrage von Datenquellen in SQL zu unterstützen, können Sie den `DataSource`-Befehl verwenden. Mit dieser Syntax können Sie Datenquellen abfragen, indem Sie sie im Befehl angeben. `FROM` Sie können bis zu 10 Datenquellen angeben.

Syntax

```
`dataSource(  
    ['DataSource1', 'DataSource2', ... 'DataSourcen']
```

```
)`
```

Beispielabfrage

```
SELECT DS1.Column1, DS1.Column2 from `dataSource(  
    ['DataSource1', 'DataSource2']  
)` as DS1 WHERE DS1.Column1 = 'ABC'
```

Umfang der Abfrage

In der AWS CLI AND-API können Sie mithilfe der Protokollgruppe, der Datenquelle und des Typs sowie der Feldindizes angeben, welche Protokolle abgefragt werden sollen.

Protokollgruppe

Die Quellenauswahl für die Protokollgruppe kann verwendet werden, wenn Kunden wissen, welche Protokollgruppe (n) genau durchsucht werden müssen

```
SELECT * FROM `logGroups(logGroupIdentifier: ['/aws/lambda/my-function'])`;
```

Datenquelle und Typ

Kunden können ihre Protokolle anhand des Datenquellennamens und des Datenquellentyps abfragen.

Die auf Datenquellen und Typen basierende Quellenauswahl kann verwendet werden, wenn Kunden wissen, welche Datenquellen genau abgefragt werden müssen. Diese Abfrage wird über eine oder mehrere Protokollgruppen ausgeführt, die die angegebene Datenquelle und den angegebenen Datentyp enthalten.

Um den Anwendungsfall für die Abfrage von Datenquellen in SQL zu unterstützen, können Sie den DataSource-Befehl verwenden. Mit dieser Syntax können Sie Datenquellen abfragen, indem Sie sie im FROM-Befehl angeben. Sie können bis zu 10 Datenquellen angeben.

Syntax:

```
`dataSource(  
    ['DataSource1.Type1', 'DataSource2.Type2', ... 'DataSourcen.TypeN']  
)`
```

Beispielabfrage:

```
SELECT DS1.Column1, DS1.Column2 from `dataSource(  
    ['DataSource1.Type1', 'DataSource2.Type2']  
)` as DS1 WHERE DS1.Column1 = 'ABC'
```

Weitere Informationen zur Abfrage nach Datenquellen finden Sie unter [Verwenden Sie Facetten, um Logs zu gruppieren und zu untersuchen](#).

Kombiniertes Beispiel

Kunden können alle Operatoren zur Quellenauswahl innerhalb der Backticks in beliebiger Reihenfolge angeben. Die Ergebnisse würden dann auf dem Schnittpunkt aller angewandten Bedingungen basieren.

Beispielsweise könnte `/aws/lambda/my-function-1` mehrere Datenquellen und -typen, einschließlich einer Vielzahl von Indizes, enthalten. Wenn die folgende Abfrage ausgeführt wurde, enthielten die zurückgegebenen Ergebnisse nur Ereignisse der Quelle und des Typs `DataSource1.Type1`, die den Kriterien „Status“ = 200 entsprechen.

```
SELECT * FROM `  
    logGroups(logGroupIdentifier: ['/aws/lambda/my-function'])  
    filterIndex('status' = 200)  
    dataSource(['DataSource1.Type1'])  
`;  
`;
```

Feldindizes

Die Auswahl der Index-based Feldquelle identifiziert automatisch relevante Protokollgruppen, wenn Ihre Filter auf indizierte Felder abzielen, wodurch das Scanvolumen und die Abfragelaufzeit reduziert werden.

Wird verwendet, `filterIndex` um nur indizierte Daten zurückzugeben, indem eine Abfrage erzwungen wird, nur Protokollgruppen zu scannen, die in einem Feld indexiert sind, das Sie in der Abfrage angeben. Für diese Protokollgruppen, die in diesem Feld indexiert sind, wird die Abfrage weiter optimiert, indem die Protokollgruppen übersprungen werden, die keine Protokollereignisse enthalten, die das in der Abfrage für das indizierte Feld angegebene Feld enthalten. Das gescannte Volumen wird weiter reduziert, indem versucht wird, nur Protokollereignisse aus diesen Protokollgruppen zu scannen, die dem in der Abfrage für diesen Feldindex angegebenen Wert

entsprechen. Weitere Informationen zu Feldindizes und deren Erstellung finden Sie unter [Feldindizes erstellen, um die Abfrageleistung zu verbessern und das Scanvolumen zu reduzieren](#).

In SQL wird `FilterIndex` verwendet, um anzugeben, welche Schlüssel-Wert-Paare als Indizes behandelt werden sollen. Die Syntax lautet wie folgt

```
SELECT * FROM `filterIndex('region' = 'us-east-1')`;
```

wobei

1. `FilterIndex (...)` gibt an, dass die darin enthaltenen Schlüsselwerte als Feldindizes behandelt werden. Jedes Schlüsselwertpaar ist durch ein Komma getrennt (Beispiel unten)
2. `'region' = 'us-east-1'` gibt die tatsächliche Bedingung an, die angewendet werden soll
 - a. Hinweis: Anstelle von `=` können Kunden `IN` verwenden, um mehrere Werte anzugeben (Beispiel unten)

Die Verwendung mehrerer `FilterIndex`-Werte würde die Bedingungen mit „UND“ kombinieren. In diesem Beispiel würden Logs abgefragt, die mit `status=200` und der Region in `us-east-1` oder `us-west-2` übereinstimmen.

```
SELECT * FROM `filterIndex('status' = 200, 'region' IN ['us-east-1', 'us-west-2'])`;
```

Einschränkungen

Die folgenden Einschränkungen gelten, wenn Sie SQL für Abfragen in Logs Insights verwenden. OpenSearch CloudWatch

- Sie können nur einen `JOIN` in eine `SELECT`-Anweisung aufnehmen.
- Sie können `JOIN` oder Unterabfragen nicht mit Datenquellenabfragen verwenden.
- Es wird nur eine Ebene verschachtelter Unterabfragen unterstützt.
- Abfragen mit mehreren Anweisungen, die durch Semikolons (;) getrennt sind, werden nicht unterstützt.
- Abfragen mit Feldnamen, die identisch sind, sich aber nur in der Groß- und Kleinschreibung unterscheiden (z. B. `field1` und `FIELD1`), werden nicht unterstützt.

Die folgende Abfrage wird beispielsweise nicht unterstützt:


```
Select AWSAccountId, AwsAccountId from LogGroup
```

Die folgende Abfrage wird jedoch unterstützt, da der Feldname (@logStream) in beiden Protokollgruppen identisch ist:

```
Select a.`@logStream`, b.`@logStream` from Table A INNER Join Table B on a.id = b.id
```

- Funktionen und Ausdrücke müssen mit Feldnamen arbeiten und Teil einer SELECT-Anweisung mit einer in der FROM-Klausel angegebenen Protokollgruppe sein.

Diese Abfrage wird beispielsweise nicht unterstützt:

```
SELECT cos(10) FROM LogGroup
```

Diese Abfrage wird unterstützt:

```
SELECT cos(field1) FROM LogGroup
```

- Wenn Sie SQL- oder PPL-Befehle verwenden, schließen Sie bestimmte Felder in Backticks ein, um sie erfolgreich abzufragen. Backticks sind für Felder mit Sonderzeichen (nicht alphabetisch und nicht numerisch) erforderlich. Schließen Sie beispielsweise, und in Backticks @message einOperation.Export.Test::Field Sie müssen Felder mit rein alphabetischen Namen nicht in Backticks einschließen.

Beispielabfrage mit einfachen Feldern:

```
SELECT SessionToken, Operation, StartTime FROM `LogGroup-A`  
LIMIT 1000;
```

Ähnliche Abfrage mit angehängten Backticks:

```
SELECT `@SessionToken`, `@Operation`, `@StartTime` FROM `LogGroup-A` LIMIT 1000;
```

Verwenden Sie natürliche Sprache, um CloudWatch Logs Insights-Abfragen zu generieren und zu aktualisieren

CloudWatch Logs unterstützt eine Abfragefunktion in natürlicher Sprache, mit der Sie Abfragen für [CloudWatch Logs Insights](#), [OpenSearch Service PPL](#), [OpenSearch Service SQL](#) und generieren und [CloudWatch Metrics Insights](#) aktualisieren können.

Mit dieser Funktion können Sie Fragen zu den CloudWatch Logs-Daten, nach denen Sie suchen, stellen oder diese in einfachem Englisch beschreiben. Die Funktion für natürliche Sprache generiert eine Abfrage auf der Grundlage einer von Ihnen eingegebenen Eingabeaufforderung und erklärt Zeile für Zeile, wie die Abfrage funktioniert. Sie können Ihre Abfrage auch aktualisieren, um Ihre Daten weiter zu untersuchen.

Abhängig von Ihrer Umgebung können Sie Eingabeaufforderungen wie „Was sind die 100 wichtigsten Quell-IP-Adressen nach übertragenen Byte?“ eingeben. und „Finde die 10 langsamsten Lambda-Funktionsanfragen.“

Note

Bei der Funktion für Abfragen in natürlicher Sprache handelt es sich um einen regionalen Dienst. In einigen Regionen führt die Funktion regionsübergreifende Aufrufe an Regionen in den USA aus, um die Abfrage-Prompts zu verarbeiten. Weitere Informationen finden Sie unter [Amazon CloudWatch erweitert die regionale Unterstützung für die Zusammenfassung und Generierung von Abfragen in natürlicher Sprache](#).

Um eine CloudWatch Logs Insights-Abfrage mit dieser Funktion zu generieren, öffnen Sie den CloudWatch Logs Insights-Abfrage-Editor, wählen Sie die Protokollgruppe aus, die Sie abfragen möchten, und wählen Sie Abfrage generieren.

Important

Um die Abfragefunktion in natürlicher Sprache verwenden zu können, müssen Sie mit den [CloudWatchLogsFullAccess](#), [CloudWatchLogsReadOnlyAccessAdministratorAccess](#), oder [ReadOnlyAccess](#) IAM-Richtlinien angemeldet sein oder über die `cloudwatch:GenerateQuery` entsprechende Berechtigung verfügen.

Beispielabfragen

In den Beispielen in diesem Abschnitt wird beschrieben, wie Abfragen mithilfe der natürlichen Sprachfunktion generiert und aktualisiert werden.

Note

Weitere Informationen zum CloudWatch Logs Insights-Abfrage-Editor und zur Syntax finden Sie unter [CloudWatch Logs Insights-Abfragesyntax](#).

Beispiele: Generieren Sie eine Abfrage in natürlicher Sprache

Um eine Abfrage in natürlicher Sprache zu generieren, geben Sie eine Aufforderung ein und wählen Sie Neue Abfrage generieren. Dieses Beispiel zeigt Abfragen, die eine einfache Suche durchführen.

Prompt

Im Folgenden finden Sie ein Beispiel für eine Eingabeaufforderung, die die Fähigkeit anweist, nach den 10 langsamsten Lambda-Funktionsaufrufen zu suchen.

```
Find the 10 slowest requests
```

Query

Die folgende Abfrage verwendet die CloudWatch Logs Insights-Abfragesprache, die von der Funktion in natürlicher Sprache auf der Grundlage der Aufforderung generiert wurde. Beachten Sie, wie die Aufforderung in einem Kommentar vor der Abfrage erscheint. Nach der Abfrage können Sie eine Erklärung lesen, in der beschrieben wird, wie die Abfrage funktioniert.

```
# Find the 10 slowest requests
fields @timestamp, @message, @duration
| sort @duration desc
| limit 10
# This query retrieves the timestamp, message and duration fields from the logs and
sorts them in descending order by duration to find the 10 slowest requests.
```

 Note

Verwenden Sie das Zahnradsymbol in Ihrem Editor, um das Erscheinungsbild Ihrer Aufforderung und die Erläuterung der Funktionsweise der Abfrage zu deaktivieren.

Prompt

Um eine OpenSearch SQL-Abfrage zu generieren, wählen Sie die Registerkarte OpenSearch SQL aus und öffnen Sie dann das Eingabeaufforderungsfeld für den Abfragegenerator, um Ihre Eingabeaufforderung in natürlicher Sprache einzugeben. Im Folgenden finden Sie ein Beispiel für eine Eingabeaufforderung, bei der die Funktion natürlicher Sprache verwendet wird, um eine OpenSearch SQL-Abfrage zu generieren.

Give me the number of errors and exceptions per hour

Query

Im Folgenden finden Sie die von dieser Aufforderung generierte SQL-Abfrage, mit der Sie die Anzahl der pro Stunde aggregierten Fehler und Ausnahmen ermitteln können:

```
SELECT DATE_FORMAT(`@timestamp`, 'yyyy-MM-dd HH') AS hour,
       COUNT(*) AS error_count
FROM `/aws/lambda/CloudWatchOdysseyQueryGen`
WHERE `@message` LIKE '%error%'
      OR `@message` LIKE '%exception%'
GROUP BY DATE_FORMAT(`@timestamp`, 'yyyy-MM-dd HH')
ORDER BY hour
```

Prompt

Um eine OpenSearch PPL-Abfrage zu generieren, wählen Sie die Registerkarte PPL aus und öffnen Sie dann das OpenSearch Eingabeaufforderungsfeld für den Abfragegenerator, um Ihre Eingabeaufforderung in natürlicher Sprache einzugeben. Im Folgenden finden Sie ein Beispiel für eine Aufforderung, bei der die Funktion natürlicher Sprache verwendet wird, um eine OpenSearch PPL-Abfrage zu generieren.

Give me all unique exception messages

Query

Im Folgenden finden Sie die PPL-Abfrage, die durch diese Aufforderung generiert wurde und die Sie verwenden können, um die eindeutigen Ausnahmemeldungen in Ihren Protokollen zu finden:

```
dedup @message  
| fields @message
```

Beispiel: Eine Abfrage in natürlicher Sprache aktualisieren

Sie können eine Abfrage aktualisieren, indem Sie die erste Eingabeaufforderung bearbeiten und dann Abfrage aktualisieren wählen.

Aktualisierte Eingabeaufforderung

Das folgende Beispiel zeigt eine aktualisierte Version der vorherigen Eingabeaufforderung. Anstatt einer Aufforderung, die nach den 10 langsamsten Lambda-Funktionsaufrufen sucht, leitet diese Aufforderung nun die Fähigkeit an, nach den 20 langsamsten Lambda-Funktionsaufrufen zu suchen und eine weitere Spalte für zusätzliche Protokollereignisse einzufügen.

```
Show top 20 slowest requests instead and display requestId as a column
```

Aktualisierte Abfrage

Im Folgenden finden Sie ein Beispiel für die aktualisierte Abfrage, die die Logs Insights-Abfragesprache verwendet. CloudWatch Beachten Sie, wie die Eingabeaufforderung in einem Kommentar vor der Abfrage erscheint. Nach der Abfrage können Sie eine Erklärung lesen, in der beschrieben wird, wie die ursprüngliche Abfrage aktualisiert wurde.

```
# Show top 20 slowest requests instead and display requestId as a column  
fields @timestamp, @message, @requestId, @duration  
| sort @duration desc  
| limit 20  
# This query modifies the original query by replacing the @message field with the  
  @requestId field and changing the limit from 10 to 20 to return the top 20 log events  
  by duration instead of the top 10.
```

Abmeldung von der Verwendung Ihrer Daten zur Serviceverbesserung

Die Eingabeaufforderungs-Daten in natürlicher Sprache, die Sie bereitstellen, um das KI-Modell zu trainieren und relevante Abfragen zu generieren, werden ausschließlich zur Bereitstellung und Wartung Ihres Services verwendet. Diese Daten könnten verwendet werden, um die Qualität von CloudWatch Logs Insights zu verbessern. Ihr Vertrauen, Ihre Privatsphäre sowie die Sicherheit Ihrer Inhalte sind unsere obersten Prioritäten. Weitere Informationen finden Sie unter [AWS -Service-Bedingungen](#) und [AWS verantwortliche KI-Richtlinie](#).

Sie können die Verwendung Ihrer Inhalte zur Entwicklung oder Verbesserung der Qualität von Abfragen in natürlicher Sprache deaktivieren, indem Sie eine Opt-Out-Richtlinie für KI-Services erstellen. Um die Datenerfassung für alle CloudWatch Logs AI-Funktionen, einschließlich der Funktion zur Abfragegenerierung, abzulehnen, müssen Sie eine Opt-Out-Richtlinie für CloudWatch Logs erstellen. Weitere Informationen finden Sie unter [Opt-Out-Richtlinien für KI-Services](#) im Benutzerhandbuch für AWS Organizations .

Unterstützte Protokolle und erkannte Felder

CloudWatch Logs Insights unterstützt verschiedene Protokolltypen. Für jedes Protokoll, das an eine Protokollgruppe der Standardklasse in Amazon CloudWatch Logs gesendet wird, generiert CloudWatch Logs Insights automatisch fünf Systemfelder:

- `@message` enthält das rohe, unverarbeitete Protokollereignis. Dies entspricht dem `message` Feld in [InputLogevent](#).
- `@timestamp` enthält den Ereignis-Zeitstempel im `timestamp`-Feld des Protokollereignisses. Dies entspricht dem `timestamp` Feld in [InputLogevent](#).
- `@ingestionTime` enthält die Uhrzeit, zu der CloudWatch Logs das Protokollereignis empfangen hat.
- `@logStream` enthält den Namen des Protokoll-Streams, zu dem das Protokollereignis hinzugefügt wurde. Protokolldatenstromgruppen werden nach demselben Prozess protokolliert, der sie generiert hat.
- `@log` ist ein Protokollgruppen-Bezeichner im folgenden Format: *account-id:log-group-name*. Beim Abfragen mehrerer Protokollgruppen kann dies nützlich sein, um zu bestimmen, zu welcher Protokollgruppe ein bestimmtes Ereignis gehört.
- `@entity` enthält abgeflachte JSON-Daten, die sich auf Entitäten beziehen, für die [Telemetriefunktion im Zusammenhang mit Explore](#).

Diese JSON-Datei kann beispielsweise eine Entität darstellen.

```
{
  "Entity": {
    "KeyAttributes": {
      "Type": "Service",
      "Name": "PetClinic"
    },
    "Attributes": {
      "PlatformType": "AWS::EC2",
      "EC2.InstanceId": "i-1234567890123"
    }
  }
}
```

Für diese Entität wären die extrahierten Systemfelder die folgenden:

```
@entity.KeyAttributes.Type = Service
@entity.KeyAttributes.Name = PetClinic
@entity.Attributes.PlatformType = AWS::EC2
@entity.Attributes.EC2.InstanceId = i-1234567890123
```

Note

Die Felderkennung wird nur für Protokollgruppen in der Standard-Protokollklasse unterstützt. Weitere Hinweise zu Protokollklassen finden Sie unter [Klassen protokollieren](#).

CloudWatch Logs Insights fügt das @-Symbol am Anfang der generierten Felder ein.

Bei vielen Protokolltypen erkennt CloudWatch Logs auch automatisch die in den Protokollen enthaltenen Protokollfelder. Diese automatischen Suchfelder sind in der folgenden Tabelle dargestellt.

Bei anderen Protokolltypen mit Feldern, die CloudWatch Logs Insights nicht automatisch erkennt, können Sie den `parse` Befehl verwenden, um extrahierte Felder für die Verwendung in dieser Abfrage zu extrahieren und zu erstellen. Weitere Informationen finden Sie unter [CloudWatch Abfragesyntax in Logs Insights](#).

Wenn der Name eines erkannten Protokollfeldes mit dem @ Zeichen beginnt, zeigt CloudWatch Logs Insights es mit einem zusätzlichen Zeichen an, das am Anfang @ angehängt wird. Wenn z. B. ein Protokollfeldname @example.com lautet, wird dieser Feldname als @@example.com angezeigt.

Note

Mit Ausnahme von @message @timestamp@log, oder können Sie Feldindizes für entdeckte Felder erstellen. Weitere Hinweise zu Feldindizes finden Sie unter [Erstellen Sie Feldindizes, um die Abfrageleistung zu verbessern und das Scanvolumen zu reduzieren](#)

Protokolltyp	Erkannte Protokollfelder
Amazon VPC-Flussprotokolle	@timestamp , @logStream , @message, accountId , endTime, interfaceId , logStatus , startTime , version, action, bytes, dstAddr, dstPort, packets, protocol, srcAddr, srcPort
Route-53-Protokolle	@timestamp , @logStream , @message, edgeLocation , ednsClientSubnet , hostZoneId , protocol, queryName , queryTimestamp , queryType , resolverIp , responseCode , version
Lambda-Protokolle	@timestamp , @logStream , @message, @requestId , @duration, @billedDuration , @type, @maxMemoryUsed , @memorySize Wenn eine Lambda-Protokollzeile eine X-Ray Trace-ID enthält, enthält sie auch die folgenden Felder: @xrayTraceId und @xraySegmentId . CloudWatch Logs Insights erkennt automatisch Protokollfelder in Lambda-Protokollen, jedoch nur für das erste eingebettete JSON-Fragment in jedem Protokollereignis. Wenn ein Lambda-Protokollereignis mehrere JSON-Fragmente enthält, können Sie die Protokollfelder mit dem parse-Befehl analysieren und extrahieren. Weitere Informationen finden Sie unter Felder in JSON-Protokollen.
CloudTrail Logs Protokolle im JSON-Format	Weitere Informationen finden Sie unter Felder in JSON-Protokollen .

Protokolltyp	Erkannte Protokollfelder
Andere Protokolltypen	@timestamp , @ingestionTime , @logStream , @message, @log.

Felder in JSON-Protokollen

Mit CloudWatch Logs Insights verwenden Sie die Punktnotation zur Darstellung von JSON-Feldern. Dieser Abschnitt enthält ein Beispiel für ein JSON-Ereignis und ein Code-Snippet, die Ihnen zeigen, wie Sie mit der Punktnotation auf JSON-Felder zugreifen können.

Example: JSON event (Beispiel: JSON-Ereignis)

```
{
  "eventVersion": "1.0",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn: aws: iam: : 123456789012: user/Alice",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "accountId": "123456789012",
    "userName": "Alice"
  },
  "eventTime": "2014-03-06T21: 22: 54Z",
  "eventSource": "ec2.amazonaws.com",
  "eventName": "StartInstances",
  "awsRegion": "us-east-2",
  "sourceIpAddress": "192.0.2.255",
  "userAgent": "ec2-api-tools1.6.12.2",
  "requestParameters": {
    "instancesSet": {
      "items": [
        {
          "instanceId": "i-abcde123"
        }
      ]
    }
  },
  "responseElements": {
    "instancesSet": {
      "items": [
```

```
{
  "instanceId": "i-abcde123",
  "currentState": {
    "code": 0,
    "name": "pending"
  },
  "previousState": {
    "code": 80,
    "name": "stopped"
  }
}
```

Das JSON-Beispielereignis enthält ein Objekt namens `userIdentity`. `userIdentity` enthält ein Feld namens `type`. Um den Wert von `type` mit Punktnotation darzustellen, verwenden Sie `userIdentity.type`.

Das JSON-Beispielereignis enthält Arrays, die sich zu Listen mit verschachtelten Feldnamen und Werten vereinfachen lassen. Verwenden Sie `requestParameters.instancesSet.items.0.instanceId`, um den Wert von `instanceId` für das erste Element in `requestParameters.instancesSet` darzustellen. Die Zahl `0`, die vor dem Feld `instanceId` platziert ist, bezieht sich auf die Position von Werten für das Feld `items`. Das folgende Beispiel enthält ein Code-Snippet, das zeigt, wie Sie in einem JSON-Protokollereignis auf verschachtelte JSON-Felder zugreifen können.

Beispiel: Abfrage

```
fields @timestamp, @message
| filter requestParameters.instancesSet.items.0.instanceId="i-abcde123"
| sort @timestamp desc
```

Der Code-Ausschnitt zeigt eine Abfrage, die Punktnotation mit dem Befehl `filter` verwendet, um auf den Wert des verschachtelten JSON-Felds `instanceId` zuzugreifen. Die Abfrage filtert nach Nachrichten, bei denen der Wert von `instanceId` `"i-abcde123"` entspricht, und gibt alle Protokollereignisse zurück, die den angegebenen Wert enthalten.

 Note

CloudWatch Logs Insights kann maximal 200 Protokollereignisfelder aus einem JSON-Protokoll extrahieren. Zusätzliche Felder, die nicht extrahiert werden, können Sie mit dem Befehl `parse` aus dem nicht analysierten Protokollereignis im Nachrichtenfeld extrahieren. Weitere Informationen zu dem `parse` Befehl finden Sie unter [Abfragesyntax](#) im CloudWatch Amazon-Benutzerhandbuch.

Erstellen Sie Feldindizes, um die Abfrageleistung zu verbessern und das Scanvolumen zu reduzieren

Sie können Feldindizes der Felder in Ihren Protokollereignissen erstellen, um effiziente, gleichheitsorientierte Suchen zu ermöglichen. Wenn Sie dann einen Feldindex in einer CloudWatch Logs Insights-Abfrage verwenden, versucht die Abfrage, die Verarbeitung von Protokollereignissen zu überspringen, von denen bekannt ist, dass sie das indizierte Feld nicht enthalten. Dadurch wird das Scanvolumen Ihrer Abfragen, die Feldindizes verwenden, reduziert, sodass Ergebnisse schneller zurückgegeben werden können. Auf diese Weise können Sie schnell Petabyte an Gesamtprotokollen in Tausenden von Protokollgruppen durchsuchen und die relevanten Protokolle schneller finden. Felder, die sich gut indexieren lassen, sind Felder, nach denen Sie häufig Abfragen durchführen müssen. Felder mit einer hohen Kardinalität von Werten eignen sich auch gut für Feldindizes, da eine Abfrage, die diese Feldindizes verwendet, schneller abgeschlossen wird, da dadurch die Protokollereignisse begrenzt werden, die dem Zielwert zugeordnet werden.

Nehmen wir beispielsweise an, Sie haben einen Feldindex für `requestId` erstellt. Dann jede CloudWatch Logs Insights-Abfrage für diese Protokollgruppe, die nur die Protokollereignisse enthält `requestId = value` oder `requestId IN [value, value, ...]` versucht, nur die Protokollereignisse zu verarbeiten, von denen bekannt ist, dass sie das indizierte Feld und den abgefragten Wert enthalten und für die CloudWatch Logs in der Vergangenheit einen Wert für dieses Feld erkannt hat.

Sie können Ihre Feldindizes auch nutzen, um effiziente Abfragen für eine größere Anzahl von Protokollgruppen zu erstellen. Wenn Sie den `filterIndex` Befehl in Ihrer Abfrage anstelle des `filter` Befehls verwenden, wird die Abfrage anhand ausgewählter Protokollgruppen für Protokollereignisse ausgeführt, die Feldindizes haben. Diese Abfragen können bis zu 10.000 Protokollgruppen scannen, die Sie auswählen können, indem Sie bis zu fünf Protokollgruppennamenpräfixe angeben. Wenn es sich um ein Überwachungskonto mit CloudWatch

kontenübergreifender Observability handelt, können Sie alle Quellkonten auswählen oder einzelne Quellkonten angeben, um die Protokollgruppen auszuwählen.“

Bei indizierten Feldern wird zwischen Groß- und Kleinschreibung unterschieden. Ein Feldindex von entspricht beispielsweise `RequestId` nicht einem Protokollereignis, das Folgendes enthält: `requestId`

Feldindizes werden nur für die strukturierten Protokollformate JSON und Serviceprotokolle unterstützt.

CloudWatch Logs stellt Standardfeldindizes für alle Protokollgruppen in der Standard-Protokollklasse bereit. Standardfeldindizes sind automatisch für die folgenden Felder verfügbar:

- `@logStream`
- `@aws.region`
- `@aws.account`
- `@source.log`
- `@data_source_name`
- `@data_source_type`
- `@data_format`
- `traceId`
- `severityText`
- `attributes.session.id`

CloudWatch Logs bietet auch Standardfeldindizes für bestimmte Kombinationen von Datenquellennamen und -typen. Standardfeldindizes sind automatisch für die folgenden Kombinationen von Datenquellennamen und -typ verfügbar:

Name und Typ der Datenquelle	Standard-Feldindizes
<code>amazon_vpc.flow</code>	<code>action</code>
	<code>logStatus</code>
	<code>region</code>

Name und Typ der Datenquelle	Standard-Feldindizes
	<code>flowDirection</code> <code>type</code>
<code>amazon_route53.resolver_query</code>	<code>query_type</code> <code>transport</code> <code>rcode</code>
<code>aws_waf.access</code>	<code>action</code> <code>httpRequest.country</code>
<code>aws_cloudtrail.data</code> <code>aws_cloudtrail.management</code>	<code>eventSource</code> <code>eventName</code> <code>awsRegion</code> <code>userAgent</code> <code>errorCode</code> <code>eventType</code> <code>managementEvent</code> <code>readOnly</code> <code>eventCategory</code> <code>requestId</code>

Standardfeldindizes gelten zusätzlich zu allen benutzerdefinierten Feldindizes, die Sie in Ihrer Richtlinie definieren. Standardfeldindizes werden nicht auf Ihr [Feldindexkontingent](#) angerechnet.

CloudWatch Protokolliert nur die Protokollereignisse, die nach der Erstellung einer Indexrichtlinie aufgenommen wurden. Protokollereignisse, die vor der Erstellung der Richtlinie aufgenommen wurden, werden nicht indexiert. Nachdem Sie einen Feldindex erstellt haben, bleibt jedes

übereinstimmende Protokollereignis ab dem Zeitpunkt der Aufnahme des Protokollereignisses 30 Tage lang indexiert.

Note

Wenn Sie eine Feldindexrichtlinie in einem Überwachungskonto erstellen, wird diese Richtlinie nicht für Protokollgruppen in verknüpften Quellkonten verwendet. Eine Feldindexrichtlinie gilt nur für das Konto, in dem sie erstellt wurde.

In den übrigen Themen dieses Abschnitts wird erklärt, wie Feldindizes erstellt werden. Informationen dazu, wie Sie in Ihren Abfragen auf Feldindizes verweisen, finden Sie unter [filterIndex](#) und [Filter](#)

Themen

- [Syntax und Kontingente für Feldindexe](#)
- [Erstellen Sie eine Feldindexrichtlinie auf Kontoebene](#)
- [Erstellen Sie eine Feldindexrichtlinie auf Protokollgruppenebene](#)
- [Protokollieren Sie die Gruppenauswahloptionen beim Erstellen einer Abfrage](#)
- [Auswirkungen des Löschens einer Feldindexrichtlinie](#)

Syntax und Kontingente für Feldindexe

Sie erstellen Feldindizes, indem Sie Feldindexrichtlinien erstellen. Sie können Indexrichtlinien auf Kontoebene erstellen, die für Ihr gesamtes Konto gelten, und Sie können auch Richtlinien erstellen, die nur für eine einzelne Protokollgruppe gelten. Für kontoweite Indexrichtlinien können Sie eine festlegen, die für alle Protokollgruppen im Konto gilt. Sie können auch Indexrichtlinien auf Kontoebene erstellen, die für eine Teilmenge von Protokollgruppen im Konto gelten, die anhand der Präfixe ihrer Protokollgruppennamen ausgewählt werden. Wenn Sie mehrere Richtlinien auf Kontoebene in demselben Konto haben, dürfen sich die Präfixe der Protokollgruppennamen für diese Richtlinien nicht überschneiden. Ebenso können Sie Indexrichtlinien auf Kontoebene erstellen, die für eine bestimmte Kombination aus Name und Typ der Datenquelle gelten. Pro Kombination aus Datenquellenname und Typ kann nur eine Kontorichtlinie erstellt werden.

Feldindexrichtlinien auf Protokollgruppenebene haben Vorrang vor Feldindexrichtlinien auf Kontoebene, die für die gesamte Protokollgruppe gelten (z. B. Richtlinien auf Kontoebene ohne Auswahlkriterien oder mit Auswahlkriterien, die auf dem Präfix von Protokollgruppennamen basieren).

Account-level Richtlinien, die auf Protokollereignisebene übereinstimmen (z. B. für eine bestimmte Kombination aus Name und Typ der Datenquelle), gelten zusätzlich zu Richtlinien, die der gesamten Protokollgruppe entsprechen. Wenn Sie eine Indexrichtlinie auf Protokollgruppenebene erstellen, verwendet diese Protokollgruppe keine Richtlinien auf Kontoebene, die auf Protokollgruppenebene übereinstimmen.

Bei Übereinstimmungen von Protokollereignissen mit den Namen von Feldindizes wird zwischen Groß- und Kleinschreibung unterschieden. Beispielsweise entspricht ein Feldindex von `RequestId` nicht einem Protokollereignis, das Folgendes enthält: `requestId`

Sie können über bis zu 40 Indexrichtlinien auf Kontoebene verfügen, von denen 20 Auswahlkriterien für das Präfix für Protokollgruppennamen und 20 auf Datenquellen basierende Auswahlkriterien verwenden können. Wenn Sie mehrere Indexrichtlinien auf Kontoebene nach Präfixen für Protokollgruppennamen gefiltert haben, können keine zwei von ihnen dieselben oder sich überschneidende Präfixe für Protokollgruppennamen verwenden. Wenn Sie beispielsweise eine Richtlinie so gefiltert haben, dass Gruppen protokolliert werden, die mit `beginnenmy-log`, können Sie keine andere Feldindexrichtlinie nach oder filtern lassen. `my-logpprod my-logging` Wenn Sie mehrere Indexrichtlinien auf Kontoebene haben, die nach Kombinationen von Datenquellenname und -typ gefiltert sind, kann keine zwei von ihnen denselben Datenquellennamen und -typ verwenden. Wenn Sie beispielsweise eine Richtlinie nach dem Datenquellennamen `amazon_vpc` und dem Datenquellentyp gefiltert haben, können `flow` Sie mit dieser Kombination keine weitere Richtlinie erstellen.

Wenn Sie über eine Indexrichtlinie auf Kontoebene verfügen, die keine Namenspräfixe hat und für alle Protokollgruppen gilt, kann keine andere Indexrichtlinie auf Kontoebene mit Präfixfiltern für Protokollgruppennamen erstellt werden. Sie können Indexrichtlinien auf Kontoebene erstellen, die Filter für den Namen und den Typ von Datenquellen verwenden.

Jede Indexrichtlinie hat die folgenden Kontingente und Einschränkungen:

- Bis zu 20 Felder können in die Richtlinie aufgenommen werden.
- Jeder Feldname kann bis zu 100 Zeichen enthalten.
- Um einen Index für ein benutzerdefiniertes Feld in Ihren Protokollgruppen zu erstellen, das mit `@` beginnt, müssen Sie das Feld mit einem zusätzlichen `@` Wert am Anfang des Feldnamens angeben. Wenn Ihre Protokollereignisse beispielsweise ein Feld mit dem Namen `enthalten@userId`, müssen Sie angeben, `@@userId` dass ein Index für dieses Feld erstellt werden soll.

Für Indexrichtlinien auf Kontoebene mit Datenquellennamen und typbasierten Auswahlkriterien gilt eine zusätzliche Einschränkung: Alle Felder müssen primitive Datentypen sein, verschachtelte Primitive werden nur für Strukturen unterstützt.

Generierte Felder und reservierte Felder

CloudWatch Logs Insights generiert bei jedem Protokollereignis automatisch Systemfelder. Diesen generierten Feldern wird das Präfix vorangestellt. @ Weitere Informationen zu generierten Feldern finden Sie unter [Unterstützte Protokolle und erkannte Felder](#).

Von diesen generierten Feldern werden die folgenden für die Verwendung als Feldindizes unterstützt:

- @logStream
- @ingestionTime
- @requestId
- @type
- @initDuration
- @duration
- @billedDuration
- @memorySize
- @maxMemoryUsed
- @xrayTraceId
- @xraySegmentId

Um diese generierten Felder zu indizieren, müssen Sie @ bei der Angabe keine zusätzlichen Felder hinzufügen, wie dies bei benutzerdefinierten Feldern der Fall ist, die mit @ beginnen. Um beispielsweise einen Feldindex für zu erstellen@logStream, geben Sie einfach @logStream als Feldindex an.

CloudWatch Logs stellt Standardfeldindizes für alle Protokollgruppen in der Standard-Protokollklasse bereit. Standardfeldindizes sind automatisch für die folgenden Felder verfügbar:

- @logStream
- @aws.region
- @aws.account

- `@source.log`
- `@data_source_name`
- `@data_source_type`
- `@data_format`
- `traceId`
- `severityText`
- `attributes.session.id`

CloudWatch Logs bietet auch Standardfeldindizes für bestimmte Kombinationen von Datenquellennamen und -typen. Standardfeldindizes sind automatisch für die folgenden Kombinationen von Datenquellennamen und -typ verfügbar:

Name und Typ der Datenquelle	Standard-Feldindizes
<code>amazon_vpc.flow</code>	<code>action</code> <code>logStatus</code> <code>region</code> <code>flowDirection</code> <code>type</code>
<code>amazon_route53.resolver_query</code>	<code>query_type</code> <code>transport</code> <code>rcode</code>
<code>aws_waf.access</code>	<code>action</code> <code>httpRequest.country</code>
<code>aws_cloudtrail.data</code>	<code>eventSource</code>
<code>aws_cloudtrail.management</code>	<code>eventName</code> <code>awsRegion</code>

Name und Typ der Datenquelle	Standard-Feldindizes
	userAgent errorCode eventType managementEvent readOnly eventCategory requestId

Standardfeldindizes gelten zusätzlich zu allen benutzerdefinierten Feldindizes, die Sie in Ihrer Richtlinie definieren. Standardfeldindizes werden nicht auf Ihr [Feldindexkontingent](#) angerechnet.

Untergeordnete Felder und Array-Felder in JSON-Protokollen

Sie können Felder indizieren, bei denen es sich um verschachtelte untergeordnete Felder oder Array-Felder in JSON-Protokollen handelt.

Sie können beispielsweise einen Index des `accessKeyId` untergeordneten Felds innerhalb des `userIdentity` Felds in diesem Protokoll erstellen:

```
{
  "eventVersion": "1.0",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn: aws: iam: : 123456789012: user/Alice",
    "accessKeyId": "11112222",
    "accountId": "123456789012",
    "userName": "Alice"
  },
  "eventTime": "2014-03-06T21: 22: 54Z",
  "eventSource": "ec2.amazonaws.com",
  "eventName": "StartInstances",
  "awsRegion": "us-east-2",
  "sourceIpAddress": "192.0.2.255",
```

```

    "userAgent": "ec2-api-tools1.6.12.2",
    "requestParameters": {
      "instancesSet": {
        "items": [{
          "instanceId": "i-abcde123",
          "currentState": {
            "code": 0,
            "name": "pending"
          },
          "previousState": {
            "code": 80,
            "name": "stopped"
          }
        }]
      }
    }
  }
}

```

Um dieses Feld zu erstellen, verweisen Sie sowohl bei der Erstellung des Feldindex als auch bei der Angabe in einer Abfrage mithilfe der Punktnotation (`userIdentity.accessKeyId`) darauf. Die Abfrage könnte wie folgt aussehen:

```

fields @timestamp, @message
| filterIndex userIdentity.accessKeyId = "11112222"

```

Im vorherigen Beispiereignis befindet sich das `instanceId` Feld in einem Array innerhalb von `requestParameters.instancesSet.items`. Um dieses Feld sowohl bei der Erstellung des Feldindex als auch bei der Abfrage darzustellen, bezeichnen Sie es als `requestParameters.instancesSet.items.0.instanceId`. Die 0 bezieht sich auf die Position dieses Felds im Array.

Dann könnte eine Abfrage für dieses Feld wie folgt aussehen:

```

fields @timestamp, @message
| filterIndex requestParameters.instancesSet.items.0.instanceId="i-abcde123"

```

Erstellen Sie eine Feldindexrichtlinie auf Kontoebene

Verwenden Sie die Schritte in diesem Abschnitt, um eine Feldindexrichtlinie zu erstellen, die für alle Protokollgruppen im Konto oder für mehrere Protokollgruppen gilt, deren Protokollgruppennamen mit derselben Zeichenfolge beginnen.

So erstellen Sie eine Feldindexrichtlinie auf Kontoebene

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im linken Navigationsbereich Einstellungen und dann die Registerkarte Protokolle aus.
3. Wählen Sie im Abschnitt Indexrichtlinien auf Kontoebene die Option Verwalten aus.
4. Wählen Sie Indexrichtlinie erstellen aus.
5. Geben Sie unter Richtlinienname einen Namen für Ihre neue Richtlinie ein.
6. Führen Sie unter Geltungsbereich der Richtlinie auswählen einen der folgenden Schritte aus:
 - Wählen Sie Alle Standardprotokollgruppen aus, damit die Indexrichtlinie auf alle Protokollgruppen der Standardklasse im Konto angewendet wird.
 - Wählen Sie Gruppen nach Präfixübereinstimmung protokollieren, um die Richtlinie auf eine Teilmenge von Protokollgruppen anzuwenden, deren Namen alle mit derselben Zeichenfolge beginnen. Geben Sie dann unter Geben Sie einen Präfixnamen ein das Präfix für diese Protokollgruppen ein.

Nachdem Sie Ihr Präfix eingegeben haben, können Sie die Option Vorschau der mit dem Präfix übereinstimmenden Protokollgruppen auswählen, um zu überprüfen, ob Ihr Präfix den erwarteten Protokollgruppen entspricht.

Wählen Sie Daten nach Datenquelle protokollieren, um die Richtlinie auf eine bestimmte Kombination aus Datenquellennamen und -typ anzuwenden. Anschließend können Sie die Datenquelle und den Datentyp aus dem Dropdownmenü auswählen.

Nachdem Sie den Namen und den Typ der Datenquelle ausgewählt haben, können Sie Felder abrufen auswählen, um den Abschnitt Feldindizes und Facetten konfigurieren mit relevanten Informationen wie den verfügbaren Feldern, den enthaltenen Protokollgruppen sowie standardmäßigen und benutzerdefinierten Feldindizes zu füllen.

7. Wählen Sie für die Konfiguration eines benutzerdefinierten Indexfeldes die Option Feldpfad hinzufügen aus, um das erste zu indizierende Feld einzugeben.

Geben Sie dann die Zeichenfolge ein, die als Wert für den Feldnamen verwendet werden soll, oder wählen Sie ein Feld aus dem Drop-down-Menü aus. Die Groß- und Kleinschreibung muss exakt mit den Angaben in den Protokollereignissen übereinstimmen. Wenn Ihre Protokollereignisse beispielsweise Folgendes beinhalten `requestId`, müssen Sie dies `requestId` hier eingeben. `RequestIdrequestID`, und `request Id` würde nicht übereinstimmen.

Wenn Sie ein benutzerdefiniertes Protokollfeld indizieren möchten, das mit dem @ Zeichen beginnt, müssen Sie bei der Eingabe der Indexzeichenfolge ein zusätzliches @ Zeichen angeben. Wenn Sie beispielsweise über ein benutzerdefiniertes Protokollfeld verfügen@emailname, geben Sie es @@emailname in das Feld Feldpfad hinzufügen ein.

Sie können auch Indizes für die @logStream Felder @ingestionTime und erstellen, die CloudWatch Logs automatisch generiert. Wenn Sie dies tun, müssen Sie @ bei der Angabe keine zusätzlichen Daten hinzufügen.

8. (Optional) Sie können nicht nur den Feldpfad angeben, sondern auch Als Facette festlegen auswählen, um das Feld als Facette zu erstellen.
9. Wiederholen Sie den vorherigen Schritt, um bis zu 20 Feldindizes hinzuzufügen.
10. Klicken Sie danach auf Erstellen.

Erstellen Sie eine Feldindexrichtlinie auf Protokollgruppenebene

Verwenden Sie die Schritte in diesem Abschnitt, um eine Feldindexrichtlinie zu erstellen, die für eine einzelne Protokollgruppe gilt.

So erstellen Sie eine Feldindexrichtlinie auf Protokollgruppenebene

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im linken Navigationsbereich Logs (Protokolle), Log groups (Protokollgruppen) aus.
3. Wählen Sie den Namen der Protokollgruppe aus.
4. Wählen Sie die Registerkarte Feldindizes.
5. Wählen Sie Feldindizes für diese Protokollgruppe verwalten
6. Wählen Sie für Feldindizes auf Protokollgruppenebene verwalten die Option Feldpfad hinzufügen aus, um das erste zu indizierende Feld einzugeben.

Geben Sie dann die Zeichenfolge ein, die als Wert für den Feldnamen verwendet werden soll. Die Groß- und Kleinschreibung muss exakt mit den Angaben in den Protokollereignissen übereinstimmen. Wenn Ihre Protokollereignisse beispielsweise Folgendes beinhaltenrequestId, müssen Sie dies requestId hier eingeben. RequestIdrequestID, und request Id würde nicht übereinstimmen.

Wenn Sie ein benutzerdefiniertes Protokollfeld indizieren möchten, das mit dem @ Zeichen beginnt, müssen Sie bei der Eingabe der Indexzeichenfolge ein zusätzliches @ Zeichen angeben.

Wenn Sie beispielsweise über ein benutzerdefiniertes Protokollfeld verfügen@emailname, geben Sie es @@emailname in das Feld Feldpfad hinzufügen ein.

Sie können auch Indizes für die @logStream Felder @ingestionTime und erstellen, die CloudWatch Logs automatisch generiert. Wenn Sie dies tun, müssen Sie @ bei der Angabe keine zusätzlichen Daten hinzufügen.

7. (Optional) Sie können nicht nur den Feldpfad angeben, sondern auch Als Facette festlegen auswählen, um das Feld als Facette zu erstellen.
8. Wiederholen Sie den vorherigen Schritt, um bis zu 20 Feldindizes hinzuzufügen.
9. Klicken Sie auf Save, sobald Sie fertig sind.

Protokollieren Sie die Gruppenauswahloptionen beim Erstellen einer Abfrage

In diesem Abschnitt werden die verschiedenen Möglichkeiten erläutert, mit denen Sie Protokollgruppen auswählen können, die in eine Abfrage aufgenommen werden sollen.

Um Protokollgruppen für eine Abfrage in der Konsole auszuwählen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Protokolle und dann Logs Insights aus.
3. Wählen Sie die Abfragesprache aus, die Sie für diese Abfrage verwenden möchten. Sie können zwischen Logs Insights QL, OpenSearchPPL oder OpenSearch SQL wählen.
4. Es gibt drei Möglichkeiten, Protokollgruppen für die Abfrage auszuwählen:
 - Verwenden Sie das Feld Name der Protokollgruppe. Dies ist die Standardauswahlmethode. Mit dieser Methode können Sie bis zu 50 Protokollgruppennamen eingeben. Wenn es sich um ein Überwachungskonto mit CloudWatch kontenübergreifender Observability handelt, können Sie Protokollgruppen sowohl in den Quellkonten als auch im Überwachungskonto auswählen. Mit einer einzigen Abfrage können Protokolle von verschiedenen Konten gleichzeitig abgefragt werden.
 - Verwenden Sie den Abschnitt Kriterien für Protokollgruppen. In diesem Abschnitt können Sie Protokollgruppen anhand des Präfixes der Protokollgruppennamen auswählen. Sie können bis zu fünf Präfixe in eine Abfrage aufnehmen. Protokollgruppen, deren Namen diese Präfixe enthalten, werden ausgewählt. Alternativ wählt die Option Alle Protokollgruppen alle Protokollgruppen aus dem Konto aus.

- Wenn es sich um ein Überwachungskonto mit CloudWatch kontenübergreifender Beobachtbarkeit handelt, können Sie im Dropdownmenü „Konto“ die Option Alle Konten auswählen, um die Protokollgruppen aus allen verknüpften Konten auszuwählen. Alternativ können Sie individuell auswählen, welche Konten für diese Abfrage berücksichtigt werden sollen.

Wenn Ihre Auswahl mit mehr als 10.000 Protokollgruppen übereinstimmt, wird eine Fehlermeldung angezeigt, die Sie auffordert, Ihre Auswahl einzuschränken.

5. Die Standard-Protokollklasse für eine Abfrage ist Standard. Sie können die Protokollklasse verwenden, um sie in Infrequent Access zu ändern.

Verwenden Sie die AWS CLI

Um diese Arten von Auswahlen zu treffen, wenn Sie eine Abfrage von der Befehlszeile aus starten, können Sie den `source` Befehl in Ihrer Abfrage verwenden. Weitere Informationen und Beispiele finden Sie unter [SOURCE](#).

Auswirkungen des Löschens einer Feldindexrichtlinie

Wenn Sie eine Feldindexrichtlinie löschen, die eine Zeit lang gültig war, passiert Folgendes:

- Bis zu 30 Tage nach dem Löschen der Richtlinie können Abfragen weiterhin von den indizierten Protokollereignissen profitieren.
- Wenn Sie eine Indexrichtlinie auf Protokollgruppenebene löschen und es bereits eine Richtlinie auf Kontoebene gibt, die für diese Protokollgruppe gelten würde, gilt die Richtlinie auf Kontoebene irgendwann auch für diese Protokollgruppe.

Verwenden Sie Facetten, um Logs zu gruppieren und zu untersuchen

Facetten sind nützlich für die Analyse von Protokollen, da Sie damit Ihre Daten interaktiv filtern und gruppieren können, ohne Abfragen ausführen zu müssen. Eine Facette ist ein Feld in Ihren Protokollen (wie `ServiceName` oder `StatusCode`), das das Filtern, Aggregieren und Analysieren über Protokollgruppen hinweg ermöglicht. Sie können die Liste der facettierten Felder in der CloudWatch Logs Insights-Konsole zusammen mit der Anzahl der Protokollereignisse für jeden Facettenwert, basierend auf dem ausgewählten Zeitraum, einsehen. Wenn Sie verschiedene

Facetten und Werte auswählen, werden die Facettenwerte und die Anzahl der Facetten in Echtzeit aktualisiert, sodass Sie Ihre Logs interaktiv erkunden können.

Jede Facette zeigt verfügbare Werte und Zählungen an, die auf Grundlage des ausgewählten Zeitraums und Abfragebereichs automatisch aus Feldern in Ihren Protokollen extrahiert und für 30 Tage aufbewahrt werden. Die angegebenen Facettenzahlen sind ungefähre Angaben. Sie können die Standardfacetten wie den Namen der Datenquelle oder den Datenquellentyp verwenden, um Ihre Protokolle zu untersuchen, oder benutzerdefinierte Facetten für jedes Feld in Ihren Protokollen erstellen. Der Datenquellenname ist ein AWS-Service oder eine AWS-Applikation, die die Protokolle generiert (z. B. Route 53, Amazon VPC oder CloudTrail), und der Datenquellentyp ist der spezifische Protokolltyp, der von diesem Service generiert wird. Standardfacetten werden von erstellt CloudWatch und beinhalten `@aws.region,@data_source_name@data_source_type`, und `@data_format`. Weitere Informationen finden Sie unter [Verwaltung von Protokollen](#). Facetten sind nur für Logs verfügbar, die in das Konto aufgenommen wurden. Wenn Sie die kontenübergreifende Observability eingerichtet haben, kann das Monitoring-Konto keine Facetten anzeigen, die auf Logs von Quellkonten basieren.

Um zusätzliche Facetten zu erstellen, wählen Sie die Felder in Ihren Protokollen aus, die für Ihre Problembehandlung relevant sind, und konfigurieren Sie sie mithilfe der Indexrichtlinien. Für benutzerdefinierte Facetten empfehlen wir, sie für Felder mit niedriger Kardinalität zu erstellen (Felder mit weniger als 100 Einzelwerten pro Tag wie Status und). ApplicationName Facetten mit mehr als 100 Einzelwerten pro Tag werden als hohe Kardinalität eingestuft und Werte für diese Facetten werden nicht angezeigt. Wählen Sie eine oder mehrere Facetten aus und wählen Sie aus, ob Abfragen in Ihren Logs ausgeführt werden sollen.

Um mit Facetten in CloudWatch Logs Insights zu beginnen:

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im Navigationsbereich Protokolle und dann Logs Insights aus.
3. (Optional) Verwenden Sie die Zeitbereichsauswahl, um einen Zeitraum auszuwählen, den Sie analysieren möchten. Für den ausgewählten Zeitraum werden die verfügbaren Facetten und Werte im Bereich angezeigt.
4. Wählen Sie Facetten aus, um Ihre Daten zu untersuchen und Aktualisierungen der Wertverteilungen zwischen den Facetten in Echtzeit zu sehen.

Facetten mit mehr als 100 Einzelwerten werden nicht angezeigt. Um bestimmte Werte abzufragen, verwenden Sie stattdessen Filter in Ihrer Abfrage.

Um eine facettenbasierte Abfrage auszuführen

1. Wählen Sie einen oder mehrere Facettenwerte aus.
2. Die Anzahl der Ereignisse wird auf der Grundlage der ausgewählten Facetten und Werte aktualisiert.
3. Wenn Facettenwerte ausgewählt werden, wird der Abfragebereich aktualisiert, um die Auswahl widerzuspiegeln.
4. Nachdem Sie die Facettenwerte ausgewählt haben, wählen Sie Ausführen, um Ihre Abfrage auszuführen.
5. Die maximale Anzahl von Einzelwerten, die pro Facette unterstützt werden, ist 100. Wenn es beispielsweise mehr als 100 Werte für eine Facette gibt, werden alle Zählungen als „-“ angezeigt, was darauf hinweist, dass die Werte unbekannt sind.

Um eine facettenbasierte Abfrage zu speichern

1. Erstellen Sie Ihre Abfrage mit einem oder mehreren Facettenwerten.
2. Die restlichen Schritte entsprechen dem Speichern einer Logs Insights-Abfrage. Weitere Informationen finden Sie unter [CloudWatch Logs Insights-Abfragen speichern](#).
3. Ihre gespeicherten Abfragen sind im Abschnitt Gespeicherte Abfragen verfügbar. Wenn Sie eine gespeicherte Abfrage abrufen, enthält sie automatisch die für die Abfrage verwendeten Facetten und Werte, sodass Sie Ihre Logs einfach analysieren können.

Um eine Facette auf Kontoebene zu erstellen

1. Um Facetten zu erstellen, müssen Sie das Feld zunächst als Index erstellen und es als Facette konfigurieren. Wählen Sie im Navigationsbereich Einstellungen, Protokolle, Indexrichtlinien auf Kontoebene aus. Alternativ können Sie im Bereich Facetten die Option Facetten verwalten auswählen.
2. Wählen Sie Neue Indexrichtlinie erstellen aus. Einzelheiten zum Erstellen von Indexrichtlinien finden Sie unter [Erstellen Sie eine Feldindexrichtlinie auf Kontoebene](#).
3. Um eine Facette zu erstellen, aktivieren Sie für das ausgewählte Feld auf der Seite zur Erstellung der Indexrichtlinie die Option Als Facet festlegen.

Facettenverwaltung mithilfe von APIs

Die Facettenverwaltung kann mithilfe der Feldindexrichtlinie erfolgen. Einzelheiten finden Sie unter [field index](#) APIs.

Feldindex-APIs

Nein.	Name	Description
1	PutIndexPolicy	Erstellt oder aktualisiert eine Feldindexrichtlinie für die spezifische Protokollgruppe
2	PutAccountPolicy	Erstellt eine Datenschutzrichtlinie auf Kontoebene, eine Abonnementfilterrichtlinie, eine Feldindexrichtlinie, eine Transformer-Richtlinie oder eine Metrikextraktionsrichtlinie, die für alle Protokollgruppen oder eine Teilmenge von Protokollgruppen im Konto gilt
3	DeleteIndexPolicy	Löscht eine Feldindexrichtlinie auf Protokollgruppenebene, die auf eine einzelne Protokollgruppe angewendet wurde
4	DeleteAccountPolicy	Löscht eine Logs-Kontorichtlinie CloudWatch

Umliegende Logs in CloudWatch Logs Insights anzeigen

Sie können Protokollereignisse anzeigen, die vor und nach einem bestimmten Protokolldatensatz aufgetreten sind, der von Ihrer CloudWatch Logs Insights-Abfrage zurückgegeben wurde.

Umgebungsprotokolle helfen Ihnen dabei, den Kontext von Fehlern, Warnungen oder anderen wichtigen Ereignissen zu verstehen.

So funktionieren Umgebungsprotokolle

Wenn Sie eine CloudWatch Logs Insights-Abfrage ausführen, enthält jeder Protokolldatensatz in den Ergebnissen die Option Umgebungsprotokolle. Wenn Sie diese Option wählen, zeigt CloudWatch Logs Insights zusätzliche Protokollereignisse aus demselben Protokollstream an, die unmittelbar vor und nach dem ausgewählten Datensatz aufgetreten sind. Diese Ereignisse werden auch dann angezeigt, wenn sie nicht mit Ihren ursprünglichen Abfragefiltern übereinstimmen.

Sie können die Anzahl der umgebenden Protokollzeilen konfigurieren, die angezeigt werden sollen. Wählen Sie zwischen 5, 10, 20, 50 oder 100 Protokollzeilen über und unter dem ausgewählten Datensatz.

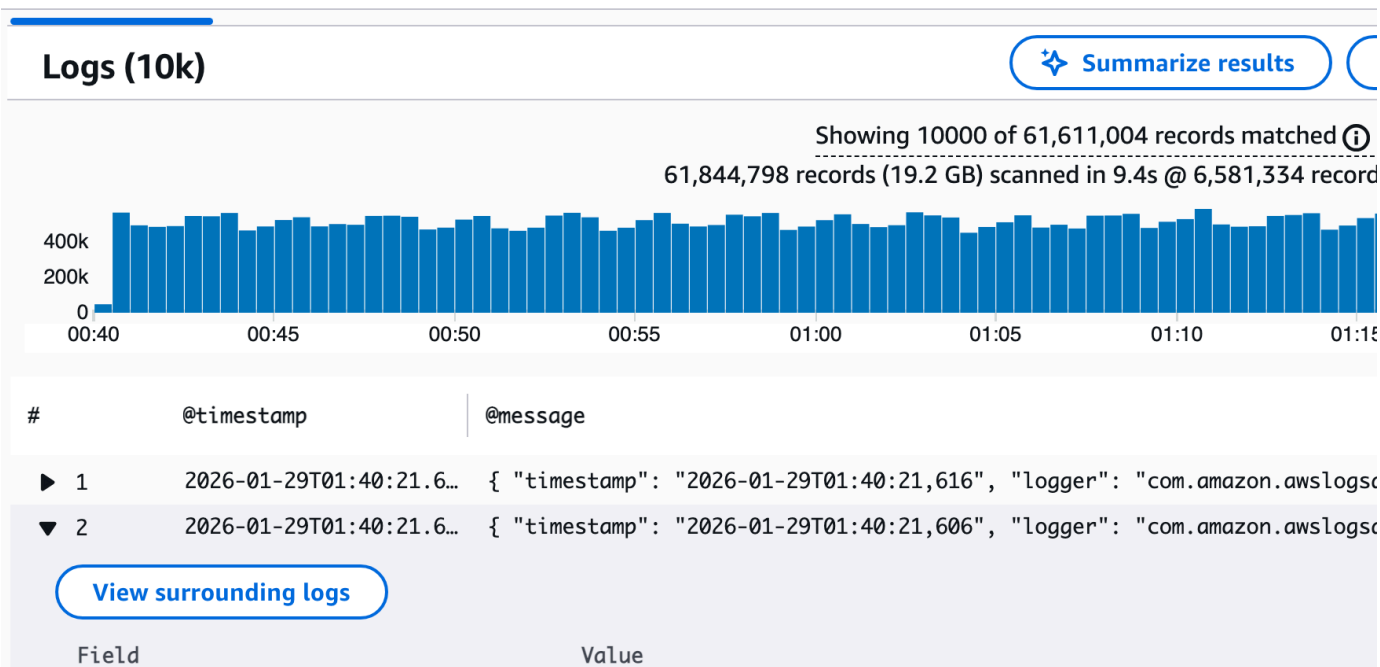
Umliegende Protokolle sind in den folgenden Szenarien nützlich:

- Debuggen von Fehlern durch Anzeige der Ereignisse, die zu einem Fehler geführt haben
- Untersuchung von Warnungen oder Timeouts in verteilten Systemen
- Verstehen Sie die Reihenfolge der Ereignisse in Ihrer Anwendung
- Analyse von Problemen in Protokollumgebungen mit hohem Volumen

Umliegende Protokolle anzeigen

Um die umliegenden Logs einzusehen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Protokolle und dann Logs Insights aus.
3. Führen Sie Ihre Abfrage aus.
4. Suchen Sie in den Abfrageergebnissen den Protokolldatensatz, den Sie untersuchen möchten.
5. Wählen Sie Umliegende Protokolle aus.



6. Wählen Sie in der Dropdownliste Anzahl der Ereignisse die Anzahl der Protokollereignisse aus, die über und unter dem ausgewählten Datensatz angezeigt werden sollen. Sie können 5, 10, 20, 50 oder 100 wählen.

Surrounding Logs [ApplicationLogs-ip-10-0-42-156.ec2.internalApplicationLogs](#) ✕

Q Search log messages... Log event count +/- 5 ▲

Timestamp	Message
▶ 2026-01-29T01:40:21.246Z	{ "timestamp": "2026-01-29T01:40:21,246", "logger": "com.amazon.awslogsanalysisdataplane.util..." }
▶ 2026-01-29T01:40:21.246Z	{ "timestamp": "2026-01-29T01:40:21,246", "logger": "com.amazon.awslogsanalysisdataplane.util..." }
▶ 2026-01-29T01:40:21.577Z	{ "timestamp": "2026-01-29T01:40:21,577", "logger": "com.amazon.awslogsanalysisdataplane.util..." }
▶ 2026-01-29T01:40:21.587Z	{ "timestamp": "2026-01-29T01:40:21,587", "logger": "com.amazon.awslogsanalysisdataplane.util..." }
▶ 2026-01-29T01:40:21.597Z	{ "timestamp": "2026-01-29T01:40:21,597", "logger": "com.amazon.awslogsanalysisdataplane.util..." }
▶ » 2026-01-29T01:40:21.607Z	{ "timestamp": "2026-01-29T01:40:21,606", "logger": "com.amazon.awslogsanalysisdataplane.util..." }
▶ 2026-01-29T01:40:21.616Z	{ "timestamp": "2026-01-29T01:40:21,616", "logger": "com.amazon.awslogsanalysisdataplane.util..." }
▶ 2026-01-29T01:40:21.625Z	{ "timestamp": "2026-01-29T01:40:21,625", "logger": "com.amazon.awslogsanalysisdataplane.util..." }
▶ 2026-01-29T01:40:21.635Z	{ "timestamp": "2026-01-29T01:40:21,635", "logger": "com.amazon.awslogsanalysisdataplane.util..." }
▶ 2026-01-29T01:40:21.644Z	{ "timestamp": "2026-01-29T01:40:21,644", "logger": "com.amazon.awslogsanalysisdataplane.util..." }
▶ 2026-01-29T01:40:21.653Z	{ "timestamp": "2026-01-29T01:40:21,653", "logger": "com.amazon.awslogsanalysisdataplane.util..." }

There are newer logs to load. [Load more](#)

Sucht in den umliegenden Protokollen

Nachdem Sie das Fenster mit den umliegenden Protokollen geöffnet haben, können Sie nach bestimmten Schlüsselwörtern suchen, um den relevanten Kontext zu finden.

Um in den umliegenden Protokollen zu suchen

1. Geben Sie im Bereich der umliegenden Protokolle Ihren Suchbegriff in das Suchfeld ein.
2. Überprüfen Sie die hervorgehobenen passenden Protokollzeilen.
3. Verwenden Sie die Navigationssteuerungen, um zwischen den Spielen zu wechseln.

Surrounding Logs [ApplicationLogs-ip-10-0-42-156.ec2.internalApplicationLogs](#) 🔗 ✕

🔍 33750 ✕ Log event count +/- 5 ▼

Timestamp	Message
▶ 2026-01-29T01:40:21.246Z	{ "timestamp": "2026-01-29T01:40:21,246", "logger": "com.amazon.awslogsanalysisdataplane.patt...
▼ 2026-01-29T01:40:21.246Z	{ "timestamp": "2026-01-29T01:40:21,246", "logger": "com.amazon.awslogsanalysisdataplane.patt...

```
{
  "timestamp": "2026-01-29T01:40:21,246",
  "logger": "com.amazon.awslogsanalysisdataplane.patternlibrary.PatternLibraryStoreImpl",
  "level": "INFO",
  "threadID": "33750",
  "threadName": "sdk-async-response-4-18426",
  "message": "Calling putPattern with anomalyDetectorId '80a6b9d7-daa0-44ac-bef7-3ef8662e4347'"
}
```

Musteranalyse

CloudWatch Logs Insights verwendet Algorithmen für maschinelles Lernen, um Muster zu finden, wenn Sie Ihre Logs abfragen. Ein Muster ist eine gemeinsame Textstruktur, die sich in Ihren Protokollfeldern wiederholt. Wenn Sie sich die Ergebnisse einer Abfrage ansehen, können Sie die Registerkarte Muster wählen, um sich die Muster anzusehen, die CloudWatch Logs anhand einer Stichprobe Ihrer Ergebnisse gefunden hat. Alternativ können Sie den `pattern` Befehl an Ihre Abfrage anhängen, um die Muster im gesamten Satz übereinstimmender Protokollereignisse zu analysieren.

Muster sind nützlich für die Analyse großer Protokollsätze, da eine große Anzahl von Protokollereignissen häufig zu wenigen Mustern komprimiert werden kann.

Betrachten Sie das folgende Beispiel mit drei Protokollereignissen.

```
2023-01-01 19:00:01 [INFO] Calling DynamoDB to store for resource id 12342342k124-12345
2023-01-01 19:00:02 [INFO] Calling DynamoDB to store for resource id 324892398123-12345
2023-01-01 19:00:03 [INFO] Calling DynamoDB to store for resource id 3ff231242342-12345
```

Im vorherigen Beispiel folgen alle drei Protokollereignisse einem Muster:

```
<Time-1> [INFO] Calling DynamoDB to store for resource id <ID-2>
```

Felder innerhalb eines Musters werden als Token bezeichnet. Felder, die innerhalb eines Musters variieren, z. B. eine Anforderungs-ID oder ein Zeitstempel, sind dynamische Token. Jedes dynamische Token wird repräsentiert durch `<string-number>`. Das *string* ist eine Beschreibung

des Datentyps, für den das Token steht. Das *number* zeigt, an welcher Stelle des Musters dieses Token im Vergleich zu den anderen dynamischen Tokens erscheint.

Zu den häufigsten Beispielen für dynamische Token gehören Fehlercodes, Zeitstempel und Anforderungs-IDs. Ein Tokenwert steht für einen bestimmten Wert eines dynamischen Tokens. Wenn ein dynamisches Token beispielsweise einen HTTP-Fehlercode darstellt, könnte dies ein Tokenwert sein 501.

Die Mustererkennung wird auch im CloudWatch Logs-Anomaliedetektor und in den Vergleichsfunktionen verwendet. Weitere Informationen erhalten Sie unter [Erkennung von Protokollanomalien](#) und [Vergleiche \(Diff\) mit früheren Zeitbereichen](#).

Erste Schritte mit der Musteranalyse

Die Mustererkennung wird automatisch in jeder CloudWatch Logs Insights-Abfrage durchgeführt. Bei Abfragen, die den `pattern` Befehl nicht enthalten, werden sowohl Ereignisse als auch Muster in den Ergebnissen protokolliert.

Wenn Sie den `pattern` Befehl in Ihre Abfrage aufnehmen, wird die Musteranalyse für den gesamten Satz übereinstimmender Protokollereignisse durchgeführt. Dadurch erhalten Sie genauere Musterergebnisse, aber die unverarbeiteten Protokollereignisse werden nicht zurückgegeben, wenn Sie den `pattern` Befehl verwenden. Wenn eine Abfrage keine Daten enthält `pattern`, basieren die Musterergebnisse entweder auf den ersten 1000 zurückgegebenen Protokollereignissen oder auf dem Grenzwert, den Sie in Ihrer Abfrage verwendet haben. Wenn Sie `pattern` in die Abfrage einbeziehen, werden die auf der Registerkarte Muster angezeigten Ergebnisse aus allen Protokollereignissen abgeleitet, denen die Abfrage entspricht.

Um mit der Musteranalyse in CloudWatch Logs Insights zu beginnen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Protokolle und dann Logs Insights aus.

Auf der Logs-Insights-Seite enthält der Abfrage-Editor eine Standardabfrage, die die 20 letzten Protokollereignisse zurückgibt.

3. Entfernen Sie die `| limit 20` Zeile im Abfragefeld, sodass die Abfrage wie folgt aussieht:

```
fields @timestamp, @message, @logStream, @log
| sort @timestamp desc
```

4. Wählen Sie in der Dropdownliste Protokollgruppe (n) auswählen eine oder mehrere Protokollgruppen für die Abfrage aus.
5. (Optional) Verwenden Sie die Zeitintervallauswahl, um einen Zeitraum auszuwählen, den Sie abfragen möchten.

Sie können zwischen Intervallen von 5 Minuten und 30 Minuten, Intervallen von 1 Stunde, 3 Stunden und 12 Stunden oder einem benutzerdefinierten Zeitrahmen wählen.

6. Wählen Sie Abfrage ausführen, um die Abfrage zu starten.

Wenn die Ausführung der Abfrage abgeschlossen ist, wird auf der Registerkarte „Protokolle“ eine Tabelle mit den von der Abfrage zurückgegebenen Protokollereignissen angezeigt. Über der Tabelle befindet sich eine Meldung darüber, wie viele Datensätze der Abfrage entsprachen, ähnlich der Meldung 10.000 von 71.101 übereinstimmenden Datensätzen anzeigen.

7. Wählen Sie die Registerkarte Muster.
8. In der Tabelle werden jetzt die in der Abfrage gefundenen Muster angezeigt. Da die Abfrage den `pattern` Befehl nicht enthielt, werden auf dieser Registerkarte nur die Muster angezeigt, die unter den 10.000 Protokollereignissen entdeckt wurden, die in der Tabelle auf der Registerkarte Protokolle angezeigt wurden.

Für jedes Muster werden die folgenden Informationen angezeigt:

- Das Muster, wobei jedes dynamische Token als angezeigt wird `<string-number>`. Das `string` ist eine Beschreibung des Datentyps, für den das Token steht. Das `number` zeigt, an welcher Stelle des Musters dieses Token im Vergleich zu den anderen dynamischen Tokens erscheint.
- Die Anzahl der Ereignisse, d. h. die Häufigkeit, mit der das Muster in den abgefragten Protokollereignissen auftauchte. Wählen Sie die Spaltenüberschrift „Anzahl der Ereignisse“, um die Muster nach Häufigkeit zu sortieren.
- Die Ereignisquote, d. h. der Prozentsatz der abgefragten Protokollereignisse, die dieses Muster enthalten.
- Der Schweregradtyp, der einer der folgenden sein wird:
 - FEHLER, wenn das Muster das Wort Error enthält.
 - WARN, wenn das Muster das Wort Warn, aber nicht das Wort Error enthält.
 - INFO, wenn das Muster weder Warn noch Error enthält.

Wählen Sie die Spaltenüberschrift Schweregrad aus, um die Muster nach Schweregrad zu sortieren.

9. Ändern Sie jetzt die Abfrage. Ersetzen Sie die `| sort @timestamp desc` Zeile in der Abfrage durch `| pattern @message`, sodass die vollständige Abfrage wie folgt lautet:

```
fields @timestamp, @message, @logStream, @log
| pattern @message
```

10. Wählen Sie Abfrage ausführen.

Wenn die Abfrage abgeschlossen ist, werden auf der Registerkarte Protokolle keine Ergebnisse angezeigt. Auf der Registerkarte Muster ist jedoch wahrscheinlich eine größere Anzahl von Mustern aufgeführt, je nachdem, wie viele Protokollereignisse insgesamt abgefragt wurden.

11. Unabhängig davon, ob Sie `pattern` in Ihre Abfrage aufgenommen haben, können Sie die Muster, die die Abfrage zurückgibt, genauer untersuchen. Wählen Sie dazu das Symbol in der Spalte Inspizieren für eines der Muster aus.

Das Fenster „Muster prüfen“ wird geöffnet und enthält Folgendes:

- Das Muster. Wählen Sie ein Token innerhalb des Musters aus, um die Werte dieses Tokens zu analysieren.
- Ein Histogramm, das die Anzahl der Vorkommen des Musters im abgefragten Zeitraum zeigt. Dies kann Ihnen helfen, interessante Trends zu identifizieren, z. B. eine plötzliche Zunahme des Auftretens eines Musters.
- Auf der Registerkarte Protokollbeispiele werden einige der Protokollereignisse angezeigt, die dem ausgewählten Muster entsprechen.
- Auf der Registerkarte Token-Werte werden die Werte des ausgewählten dynamischen Tokens angezeigt, sofern Sie eines ausgewählt haben.

 Note

Für jedes Token werden maximal 10 Token-Werte erfasst. Die Anzahl der Tokens ist möglicherweise nicht genau. CloudWatch Logs verwendet einen probabilistischen Zähler, um die Tokenanzahl zu generieren, nicht den absoluten Wert.

- Auf der Registerkarte Verwandte Muster werden andere Muster angezeigt, die häufig fast zur gleichen Zeit wie das Muster aufgetreten sind, das Sie untersuchen. Wenn beispielsweise ein

Muster für eine ERROR Nachricht normalerweise von einem anderen Protokollereignis begleitet wurde, das als INFO mit zusätzlichen Details gekennzeichnet war, wird dieses Muster hier angezeigt.

Details zum Pattern-Befehl

Dieser Abschnitt enthält weitere Informationen über den `pattern` Befehl und seine Verwendung.

- Im vorherigen Tutorial haben wir den `sort` Befehl entfernt, als wir ihn hinzugefügt haben `pattern`, weil eine Abfrage nicht gültig ist, wenn sie nach einem `pattern` Befehl einen `sort` Befehl enthält. Es ist gültig, ein `pattern` vor einem `sort` zu haben.

Weitere Informationen zur `pattern` Syntax finden Sie unter [pattern](#).

- Wenn Sie `pattern` in einer Abfrage verwenden, `@message` muss eines der Felder im `pattern` Befehl ausgewählt sein.
- Sie können den `filter` Befehl vor einem `pattern` Befehl einschließen, damit nur der gefilterte Satz von Protokollereignissen als Eingabe für die Musteranalyse verwendet wird.
- Um Musterergebnisse für ein bestimmtes Feld anzuzeigen, z. B. ein aus dem `parse` Befehl abgeleitetes Feld, verwenden Sie `pattern @fieldname`.
- Abfragen, bei denen es sich nicht um eine Protokollausgabe handelt, wie z. B. Abfragen mit dem `stats` Befehl, geben keine Musterergebnisse zurück.

Speichern Sie CloudWatch Logs Insights-Abfragen und führen Sie sie erneut aus

Nachdem Sie eine Abfrage erstellt haben, können Sie sie speichern, um sie später erneut auszuführen. Abfragen werden in einer Ordnerstruktur gespeichert, sodass Sie sie organisieren können. Sie können bis zu 1000 Abfragen pro Region und pro Konto speichern.

Abfragen werden auf einer Region-specific Ebene gespeichert, nicht auf einer benutzerspezifischen Ebene. Wenn Sie eine Abfrage erstellen und speichern, können andere Benutzer mit Zugriff auf CloudWatch Protokolle in derselben Region alle gespeicherten Abfragen und ihre Ordnerstrukturen in der Region sehen.

Um eine Abfrage zu speichern, müssen Sie bei einer Rolle mit der Berechtigung angemeldet sein `logs:PutQueryDefinition`. Um die Liste Ihrer gespeicherten Abfragen anzuzeigen, müssen Sie bei einer Rolle mit der Berechtigung `logs:DescribeQueryDefinitions` angemeldet sein.

Note

Sie können Abfragen mit Parametern erstellen und speichern — wiederverwendbare Vorlagen mit benannten Platzhaltern. Anstatt mehrere Varianten derselben Abfrage mit unterschiedlichen Werten zu speichern, erstellen Sie eine Vorlage und geben Sie bei der Ausführung unterschiedliche Parameterwerte an. Diese Funktion wird derzeit nur für Abfragen unterstützt, die die Logs Insights-Abfragesprache verwenden. Weitere Informationen finden Sie unter [Gespeicherte Abfragen mit Parametern verwenden](#).

Console

Um eine Abfrage zu speichern

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Logs Insights aus.
3. Erstellen Sie im Abfrage-Editor eine Abfrage.
4. Wählen Sie Speichern.
5. Geben Sie einen Namen für die Abfrage ein.
6. (Optional) Wählen Sie den Ordner aus, in dem Sie die Abfrage speichern möchten. Wählen Sie Create new (Neu erstellen) aus, um einen Ordner zu erstellen. Wenn Sie einen neuen Ordner erstellen, können Sie Schrägstriche (/) im Ordnernamen verwenden, um eine Ordnerstruktur zu definieren. Wenn Sie beispielsweise einen neuen Ordner mit **folder-level-1/folder-level-2** benennen, wird der Ordner **folder-level-1** auf der obersten Ebene erstellt. In diesem Ordner befindet sich ein weiterer Ordner namens **folder-level-2**. Die Abfrage wird in **folder-level-2** gespeichert.
7. (Optional) Ändern Sie die Protokollgruppen oder den Abfragetext der Abfrage.
8. (Optional) Gehen Sie wie folgt vor, um Parameter in Ihrer Abfrage zu verwenden:
 - a. Fügen Sie Ihrer Abfrage Parameter hinzu. Ersetzen Sie statische Werte mithilfe der `{{parameter}}` Syntax durch Platzhalter (doppelte Klammern vor und nach dem Parameternamen).

Beispiel: Ursprüngliche Abfrage mit statischen Werten:

```
fields @timestamp, @message
| filter level = "Error"
| filter applicationName = "OrderService"
```

Aktualisierte Abfrage mit Parametern:

```
fields @timestamp, @message
| filter level = {{logLevel}}
| filter applicationName = {{applicationName}}
```

b. Definieren Sie die in Ihrer Abfrage verwendeten Parameter. Geben Sie für jeden Platzhalterparameter Folgendes an:

- Name: Muss exakt mit dem Platzhalternamen übereinstimmen (z. B. `logLevel`, `applicationName`).
- Standardwert (optional): Der Wert, der verwendet werden soll, wenn kein Parameterwert angegeben wird.
- Beschreibung (optional): Erläutert den Zweck des Parameters.

c. Abfragen mit Parametern können ausgeführt werden, indem der Abfragenname mit einem \$ Präfix verwendet und die Parameternamen als Schlüssel-Wert-Paare übergeben werden. Einzelheiten finden Sie unter [So führen Sie eine gespeicherte Abfrage aus](#).

9. Wählen Sie Speichern.

AWS CLI

Um eine Abfrage zu speichern, verwenden Sie `put-query-definition`:

```
aws logs put-query-definition \
  --name "ErrorsByLevel" \
  --query-string "fields @timestamp, @message | filter level = \"ERROR\" \" \" \
  --log-group-names "/aws/lambda/my-function" \
  --region us-east-1
```

(Optional) Um eine Abfrage mit Parametern zu speichern, fügen Sie die `--parameters` Option hinzu und verwenden Sie `{{parameterName}}` Platzhalter in der Abfragezeichenfolge:

```
aws logs put-query-definition \
  --name "ErrorsByLevel" \
```

```
--query-string "fields @timestamp, @message | filter level = {{logLevel}} | filter
applicationName = {{applicationName}}" \
--parameters '[{"name":"logLevel","defaultValue":"ERROR","description":"Log level
to filter"}],
{"name":"applicationName","defaultValue":"OrderService","description":"Application
name to filter"}]' \
--log-group-names "/aws/lambda/my-function" \
--region us-east-1
```

Um eine Abfrage in einem Ordner zu speichern, stellen Sie dem Abfragenamen den Ordnerpfad voran:

```
aws logs put-query-definition \
--name "my-folder/ErrorsByLevel" \
--query-string "fields @timestamp, @message | filter level = {{logLevel}}" \
--parameters '[{"name":"logLevel","defaultValue":"ERROR","description":"Log level
to filter"}]' \
--log-group-names "/aws/lambda/my-function" \
--region us-east-1
```

API

Um eine Abfrage zu speichern, rufen Sie [PutQueryDefinition](#):

```
{
  "name": "ErrorsByLevel",
  "queryString": "fields @timestamp, @message | filter level = \"ERROR\"",
  "logGroupNames": ["/aws/lambda/my-function"]
}
```

(Optional) Um eine Abfrage mit Parametern zu speichern, schließen Sie das `parameters` Feld ein und verwenden Sie `{{parameterName}}` Platzhalter in der Abfragezeichenfolge:

```
{
  "name": "ErrorsByLevel",
  "queryString": "fields @timestamp, @message | filter level = {{logLevel}} | filter
applicationName = {{applicationName}}",
  "logGroupNames": ["/aws/lambda/my-function"],
  "parameters": [
    {
      "name": "logLevel",
```

```
{
  "defaultValue": "ERROR",
  "description": "Log level to filter"
},
{
  "name": "applicationName",
  "defaultValue": "OrderService",
  "description": "Application name to filter"
}
]
```

 Tip

Sie können einen Ordner für gespeicherte Abfragen mit `PutQueryDefinition` erstellen. Um einen Ordner für Ihre gespeicherten Abfragen zu erstellen, verwenden Sie einen Schrägstrich (/), um Ihrem gewünschten Abfragenamen den gewünschten Ordnernamen voranzustellen: `<folder-name>/<query-name>`. Weitere Informationen zu dieser Aktion finden Sie unter [PutQueryDefinition](#).

Console

So führen Sie eine gespeicherte Abfrage aus

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Logs Insights aus.
3. Wählen Sie auf der rechten Seite Queries (Abfragen) aus.
4. Wählen Sie Ihre Abfrage aus der Liste Gespeicherte Abfragen aus. Der Abfragetext wird im Abfrage-Editor angezeigt.
5. (Optional) Um eine Abfrage mit Parametern zu verwenden:
 - a. Wählen Sie im Seitenbereich Gespeicherte Abfragen das Pluszeichen (+) neben dem Abfragenamen aus.
 - b. Die Abfrage mit den Parametern wird im Abfrage-Editor angezeigt. Wenn Sie beispielsweise das Pluszeichen (+) neben wählenErrorsByLevel, wird der Abfrage-Editor mit folgenden Informationen gefüllt: `$ErrorsByLevel(level=, applicationName=)`

- c. Geben Sie die Werte für die Parameter (level, applicationName) an und führen Sie die Abfrage aus. Beispiel: `$ErrorsByLevel(level= "ERROR", applicationName= "OrderService")`
6. Klicken Sie auf Ausführen.

AWS CLI

Um eine gespeicherte Abfrage mit Parametern auszuführen

Verwenden Sie es `start-query` mit der folgenden `$QueryName()` Syntax:

```
aws logs start-query \  
  --log-group-names "/aws/lambda/my-function" \  
  --start-time 1707566400 --end-time 1707570000 \  
  --query-string '$ErrorsByLevel(level= "ERROR", applicationName= "OrderService")' \  
  --region us-east-1
```

API

Um eine gespeicherte Abfrage mit Parametern auszuführen

Rufen Sie [StartQuery](#) mit der `$QueryName()` Syntax im `queryString` Feld auf:

```
{  
  "logGroupNames": ["/aws/lambda/my-function"],  
  "startTime": 1707566400,  
  "endTime": 1707570000,  
  "queryString": "$ErrorsByLevel(level=\"ERROR\", applicationName= \"OrderService  
  \")\"  
}
```

So speichern Sie eine neue Version einer gespeicherten Abfrage

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Logs Insights aus.
3. Wählen Sie auf der rechten Seite Queries (Abfragen) aus.
4. Wählen Sie Ihre Abfrage aus der Liste Saved queries (Gespeicherte Abfragen) aus.
Anschließend wird sie im Abfrage-Editor angezeigt.

5. Ändern Sie die Abfrage. Wenn Sie die Abfrage ausführen müssen, um Ihre Arbeit zu überprüfen, wählen Sie Run query (Abfrage ausführen) aus.
6. Wenn Sie bereit sind, die neue Version zu speichern, wählen Sie Actions (Aktionen) und Save as (Speichern unter) aus.
7. Geben Sie einen Namen für die Abfrage ein.
8. (Optional) Wählen Sie den Ordner aus, in dem Sie die Abfrage speichern möchten. Wählen Sie Create new (Neu erstellen) aus, um einen Ordner zu erstellen. Wenn Sie einen neuen Ordner erstellen, können Sie Schrägstriche (/) im Ordernamen verwenden, um eine Ordnerstruktur zu definieren. Wenn Sie beispielsweise einen neuen Ordner mit **folder-level-1/folder-level-2** benennen, wird der Ordner **folder-level-1** auf der obersten Ebene erstellt. In diesem Ordner befindet sich ein weiterer Ordner namens **folder-level-2**. Die Abfrage wird in **folder-level-2** gespeichert.
9. (Optional) Ändern Sie die Protokollgruppen oder den Abfragetext der Abfrage.
10. Wählen Sie Speichern.

Um eine Abfrage zu löschen, müssen Sie bei einer Rolle mit der Berechtigung `logs:DeleteQueryDefinition` angemeldet sein.

So bearbeiten oder löschen Sie eine gespeicherte Abfrage

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Logs Insights aus.
3. Wählen Sie auf der rechten Seite Queries (Abfragen) aus.
4. Wählen Sie Ihre Abfrage aus der Liste Saved queries (Gespeicherte Abfragen) aus. Anschließend wird sie im Abfrage-Editor angezeigt.
5. Wählen Sie Actions (Aktionen), Edit (Bearbeiten) oder Actions (Aktionen), Delete (Löschen) aus.

Gespeicherte Abfragen mit Parametern verwenden

Gespeicherte Abfragen mit Parametern sind wiederverwendbare Abfragevorlagen mit benannten Platzhaltern. Anstatt mehrere Kopien fast identischer Abfragen zu verwalten, können Sie eine Vorlage speichern und bei der Ausführung der Abfrage unterschiedliche Parameterwerte angeben. Parameter werden nur in der CloudWatch Logs Insights-Abfragesprache unterstützt.

Funktionsweise

Beim Speichern einer Abfrage identifizieren Platzhalter die Werte, die Sie bei der Ausführung der Abfrage angeben können. Platzhalter verwenden die `{{parameterName}}` Syntax. Im Folgenden finden Sie ein Beispiel für eine gespeicherte Abfrage `ErrorsByLevel` mit dem Namen `logLevel` und `applicationName`.

```
fields @timestamp, @message
| filter level = {{logLevel}}
| filter applicationName = {{applicationName}}
```

Um eine gespeicherte Abfrage auszuführen, können Sie sie aufrufen, indem Sie den Abfragenamen mit dem Präfix verwenden `$` und die Parameterwerte übergeben. Die CloudWatch Logs Insights-Abfrage-Engine ersetzt jeden Platzhalter. Wenn ein Parameter Standardwerte enthält, werden diese Werte verwendet, sofern keine anderen Werte angegeben werden.

```
# Run query by using query name and passing parameter values explicitly
$ErrorsByLevel(logLevel = "WARN", applicationName = "OrderService")

# Run query without specifying parameter values - default values are used in this case.
$ErrorsByLevel()
```

Gespeicherte Abfragenamen, die Leerzeichen oder Sonderzeichen enthalten, müssen mit Backticks umschlossen werden:

```
$`Errors By Level`(logLevel = "WARN")
```

Beispiel für gespeicherte Abfragen mit Parametern

Hinzufügen eines Ergebnislimits als Parameter

Abfragename: `ErrorsByLevel` mit Parametern `logLevel` (Standard:"ERROR"), `applicationName` (Standard:"OrderService") und `maxResults` (Standard:50)

```
fields @timestamp, @message, @logStream
| filter level = {{logLevel}}
| filter applicationName = {{applicationName}}
| sort @timestamp desc
| limit {{maxResults}}
```

```
# Run the query using the query name and passing parameter values
```



```
$ErrorsByLevel(logLevel = "WARN", applicationName = "OrderService", maxResults = 100)
```

Verwenden mehrerer gespeicherter Abfragen mit Parametern

Das folgende Beispiel verwendet `ErrorsByLevel` eine zweite gespeicherte Abfrage `RecentN`, die als `sort @timestamp desc | limit {{count}}` (mit Parameter `count`, Standard `20`) definiert ist. Die CloudWatch Logs Insights-Abfrage-Engine erweitert jede Abfrage, bevor sie ausgeführt wird.

```
# Using multiple queries with parameters in sequence
$ErrorsByLevel(logLevel = "WARN", applicationName = "OrderService")
| $RecentN(count = 10)

# Each of the queries is expanded, resulting in the following query when it is run.
fields @timestamp, @message
| filter level = "WARN"
| filter applicationName = "OrderService"
| sort @timestamp desc
| limit 10
```

Kontingente und Fehlerbehandlung

Note

Jede gespeicherte Abfrage kann maximal 20 Parameter haben.

Die erweiterte Abfragezeichenfolge darf 10.000 Zeichen nicht überschreiten. Parameternamen müssen mit einem Buchstaben oder Unterstrich beginnen. Eine gespeicherte Abfrage kann nicht auf eine andere gespeicherte Abfrage verweisen (verschachtelte Aufrufe werden nicht unterstützt).

Häufige Fehler

Fehler	Ursache
Parameter werden nur für die CWLI-Abfragesprache unterstützt	Parameter werden nur in der CloudWatch Logs Insights-Abfragesprache unterstützt.

Fehler	Ursache
Erforderliche Parameter wurden nicht in QueryString gefunden	Für einen Parameternamen in <code>--parameters</code> gibt es keine Übereinstimmung <code>{{placeholder}}</code> in der Abfragezeichenfolge.
Die Anzahl der Parameter überschreitet den Höchstwert von 20	Gespeicherte Abfragen unterstützen derzeit nur 20 Parameter.
Doppelter Parametername	Die Abfragedefinition enthält doppelte Parameter in <code>parameter s</code> .

Note

Um eine gespeicherte Abfrage mit Parametern zu erstellen oder zu aktualisieren, benötigen Sie die `logs:PutQueryDefinition` entsprechende Berechtigung. Um eine auszuführen, benötigen Sie `logs:StartQuery` und `logs:DescribeQueryDefinitions`.

Abfrage zum Dashboard hinzufügen oder Abfrageergebnisse exportieren

Nachdem Sie eine Abfrage ausgeführt haben, können Sie die Abfrage zu einem CloudWatch Dashboard hinzufügen oder die Ergebnisse in die Zwischenablage kopieren.

Zu Dashboards hinzugefügte Abfragen werden bei jedem Laden des Dashboards und bei jeder Aktualisierung des Dashboards ausgeführt. Diese Abfragen werden auf Ihr Limit von 100 gleichzeitigen CloudWatch Logs Insights-Abfragen angerechnet.

So fügen Sie Abfrageergebnisse zu einem Dashboard hinzu

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Logs Insights aus.
3. Wählen Sie eine oder mehrere Protokollgruppen aus und führen Sie eine Abfrage aus.
4. Wählen Sie Add to dashboard (Zu Dashboard hinzufügen) aus.

5. Wählen Sie das Dashboard aus. Sie können auch Create new (Neu erstellen) auswählen, um ein Dashboard für die Abfrageergebnisse zu erstellen.
6. Wählen Sie den Widget-Typ aus, der für die Abfrageergebnisse verwendet werden soll.
7. Geben Sie einen Namen für das Widget ein.
8. Wählen Sie Add to dashboard (Zu Dashboard hinzufügen) aus.

So kopieren Sie Abfrageergebnisse in die Zwischenablage oder laden die Abfrageergebnisse herunter

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Logs Insights aus.
3. Wählen Sie eine oder mehrere Protokollgruppen aus und führen Sie eine Abfrage aus.
4. Wählen Sie Export results (Ergebnisse exportieren) und dann die gewünschte Option aus.

Anzeigen von laufenden Abfragen oder Abfrageverlauf

Sie können die aktuell laufenden Abfragen sowie Ihren aktuellen Abfrageverlauf einsehen.

Abfragen, die derzeit ausgeführt werden, beinhalten Abfragen, die Sie einem Dashboard hinzugefügt haben. Sie sind auf 100 gleichzeitige CloudWatch Logs Insights-Abfragen pro Konto beschränkt, einschließlich Abfragen, die zu Dashboards hinzugefügt wurden. Darüber hinaus können Sie 15 Abfragen gleichzeitig für OpenSearch Service PPL oder Service SQL ausführen. OpenSearch

So zeigen Sie Ihren aktuellen Abfrageverlauf an

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Logs Insights aus.
3. Wählen Sie Verlauf, wenn Sie das neue Design für die CloudWatch Logs-Konsole verwenden. Wenn Sie das alte Design verwenden, wählen Sie Actions (Aktionen) und dann View query history for this account (Abfrageverlauf für dieses Konto anzeigen) aus.

Eine Liste Ihrer letzten Abfragen wird angezeigt. Sie können sie erneut ausführen, indem Sie die Abfrage und dann Run (Ausführen) auswählen.

Unter Status wird für alle Abfragen, CloudWatch die gerade ausgeführt werden, der Eintrag In Bearbeitung angezeigt.

Verschlüsseln Sie Abfrageergebnisse mit AWS Key Management Service

Standardmäßig verschlüsselt CloudWatch Logs die gespeicherten Ergebnisse Ihrer CloudWatch Logs Insights-Abfragen mit der serverseitigen CloudWatch Logs-Standardverschlüsselungsmethode. Sie können sich stattdessen dafür entscheiden, diese Ergebnisse mit einem AWS KMS Schlüssel zu verschlüsseln. Wenn Sie Ihren Verschlüsselungsergebnissen einen AWS KMS Schlüssel zuordnen, verwendet CloudWatch Logs diesen Schlüssel, um die gespeicherten Ergebnisse aller Abfragen im Konto zu verschlüsseln.

Wenn Sie später die Zuordnung eines Schlüssels zu Ihren Abfrageergebnissen aufheben, kehrt CloudWatch Logs für spätere Abfragen zur Standardverschlüsselungsmethode zurück. Die Abfragen, die während der Zuordnung des Schlüssels ausgeführt wurden, sind jedoch weiterhin mit diesem Schlüssel verschlüsselt. CloudWatch Protokolle können diese Ergebnisse auch dann zurückgeben, wenn die Zuordnung des KMS-Schlüssels aufgehoben wurde, da CloudWatch Protokolle weiterhin auf den Schlüssel verweisen können. Wenn der Schlüssel jedoch später deaktiviert wird, kann CloudWatch Logs die Abfrageergebnisse, die mit diesem Schlüssel verschlüsselt wurden, nicht lesen.

Important

CloudWatch Logs unterstützt nur symmetrische KMS-Schlüssel. Verwenden Sie keinen asymmetrischen Schlüssel, um Ihre Abfrageergebnisse zu verschlüsseln. Weitere Informationen finden Sie unter [Verwenden von symmetrischen und asymmetrischen Schlüsseln](#).

Einschränkungen

- Um die folgenden Schritte ausführen zu können, benötigen Sie die folgenden Berechtigungen: `kms:CreateKey`, `kms:GetKeyPolicy` und `kms:PutKeyPolicy`.
- Nachdem Sie die Verknüpfung eines Schlüssels mit Ihren Abfrageergebnissen hergestellt oder aufgehoben haben, kann es bis zu fünf Minuten dauern, bis der Vorgang wirksam wird.
- Wenn Sie CloudWatch Logs den Zugriff auf einen zugehörigen Schlüssel entziehen oder einen zugehörigen KMS-Schlüssel löschen, können Ihre verschlüsselten Daten in CloudWatch Logs nicht mehr abgerufen werden.

- Sie können die CloudWatch Konsole nicht verwenden, um einen Schlüssel zuzuordnen. Sie müssen die AWS CLI oder CloudWatch Logs-API verwenden.

Schritt 1: Erstellen Sie ein AWS KMS key

Um einen KMS-Schlüssel zu erstellen, verwenden Sie den folgenden Befehl [create-key](#):

```
aws kms create-key
```

Die Ausgabe enthält die Schlüssel-ID und den Amazon-Ressourcennamen (ARN) des Schlüssels. Das Folgende ist eine Beispielausgabe:

```
{
  "KeyMetadata": {
    "Origin": "AWS_KMS",
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "Description": "",
    "KeyManager": "CUSTOMER",
    "Enabled": true,
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "KeyState": "Enabled",
    "CreationDate": 1478910250.94,
    "Arn": "arn:aws:kms:us-west-2:123456789012:key/6f815f63-e628-448c-8251-
e40cb0d29f59",
    "AWSAccountId": "123456789012",
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ]
  }
}
```

Schritt 2: Festlegen von Berechtigungen auf dem KMS-Schlüssel

Standardmäßig sind alle KMS-Schlüssel privat. Nur der Ressourcenbesitzer kann mit ihnen Daten verschlüsseln und entschlüsseln. Der Ressourceninhaber kann jedoch anderen Benutzern und Ressourcen Zugriffsberechtigungen für den Schlüssel erteilen. Mit diesem Schritt erteilen Sie dem Prinzipal des CloudWatch Logs-Dienstes die Erlaubnis, den Schlüssel zu verwenden. Dieser Dienstprinzipal muss sich in derselben AWS Region befinden, in der der Schlüssel gespeichert ist.

Als bewährte Methode empfehlen wir, die Verwendung des Schlüssels nur auf die von Ihnen angegebenen AWS Konten zu beschränken.

Speichern Sie zunächst die Standardrichtlinie für Ihren KMS-Schlüssel als `policy.json` ab, indem Sie den folgenden Befehl `get-key-policy` verwenden:

```
aws kms get-key-policy --key-id key-id --policy-name default --output text > ./policy.json
```

Öffnen Sie die Datei `policy.json` in einem Texteditor und fügen Sie den in Fettschrift angezeigten Abschnitt aus einer der folgenden Anweisungen hinzu. Sie trennen die vorhandene Anweisung von der neuen Anweisung durch ein Komma. In diesen Anweisungen `Condition` werden Abschnitte verwendet, um die Sicherheit des AWS KMS Schlüssels zu erhöhen. Weitere Informationen finden Sie unter [AWS KMS Schlüssel und Verschlüsselungskontext](#).

Der Condition Abschnitt in diesem Beispiel beschränkt die Verwendung des AWS KMS Schlüssels auf die CloudWatch Logs Insights-Abfrageergebnisse im angegebenen Konto.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.region.amazonaws.com"
      },
      "Action": [
        "kms:Encrypt*",
        "kms:Decrypt*",

```

```

        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:Describe"
    ],
    "Resource": "*",
    "Condition": {
        "ArnEquals": {
            "aws:SourceArn": "arn:aws:logs:us-east-1:111122223333:query-
result:*"
        },
        "StringEquals": {
            "aws:SourceAccount": "111122223333"
        }
    }
}
]
}

```

Fügen Sie abschließend die aktualisierte Richtlinie hinzu, indem Sie folgenden Befehl [put-key-policy](#) verwenden:

```
aws kms put-key-policy --key-id key-id --policy-name default --policy file://
policy.json
```

Schritt 3: Ordnen Sie Ihren Abfrageergebnissen einen KMS-Schlüssel zu

So ordnen Sie den KMS-Schlüssel den Abfrageergebnissen im Konto zu

Verwenden Sie den Befehl [disassociate-kms-key](#) wie folgt:

```
aws logs associate-kms-key --resource-identifizier "arn:aws:logs:region:account-id:query-
result:*" --kms-key-id "key-arn"
```

Schritt 4: Trennen Sie die Zuordnung eines Schlüssels zu den Abfrageergebnissen im Konto

Um die Verknüpfung eines KMS-Schlüssels mit Abfrageergebnissen aufzuheben, verwenden Sie den folgenden Befehl [disassociate-kms-key](#):

```
aws logs disassociate-kms-key --resource-identifier "arn:aws:logs:region:account-id:query-result:*
```

Generieren Sie eine Zusammenfassung in natürlicher Sprache aus den CloudWatch Logs Insights-Abfrageergebnissen

Die Analyse von Protokolldaten ist entscheidend, um das Verhalten Ihrer Anwendungen zu verstehen, aber die Interpretation großer Mengen von Protokolleinträgen kann zeitaufwändig sein. CloudWatch Logs Insights bietet jetzt eine Funktion zur Zusammenfassung in natürlicher Sprache, die komplexe Abfrageergebnisse in klare, präzise Zusammenfassungen umwandelt. Diese Funktion hilft Ihnen, Probleme schnell zu identifizieren und umsetzbare Erkenntnisse aus Ihren Protokolldaten zu gewinnen.

Funktionsweise

CloudWatch Logs Insights kann mithilfe von Amazon Bedrock eine menschenlesbare Zusammenfassung aus Ihren Abfrageergebnissen generieren. Die Funktion unterstützt alle CloudWatch Logs Insights-Abfragesprachen und bietet klare, umsetzbare Erkenntnisse aus Ihren Protokolldaten.

Regionale Verfügbarkeit und Datenverarbeitung

Important

Wenn Sie diese Funktion verwenden, werden Ihre Abfrageergebnisse möglicherweise auf andere Weise verarbeitet AWS-Region. Wenn Sie beispielsweise eine Abfrage in USA Ost (Nord-Virginia) ausführen, kann die Zusammenfassung in USA West (Oregon) erfolgen.

In der folgenden Tabelle ist die mögliche Verarbeitung AWS-Region für die verschiedenen Regionen aufgeführt, in denen die Funktion für Abfrageergebnisse verfügbar ist:

Geografie der unterstützten Logs CloudWatch	Mögliche Verarbeitungsregion
Vereinigte Staaten (USA)	Region USA Ost (Nord-Virginia)
	Region USA Ost (Ohio)

Geografie der unterstützten Logs CloudWatch	Mögliche Verarbeitungsregion
	Region USA West (Oregon)
Europa	Region Europa (Frankfurt) Europe (Ireland) Region Region Europa (Paris) Region Europa (Stockholm) Region Europa (London)
Asien-Pazifik	Region USA Ost (Nord-Virginia) Region USA Ost (Ohio) Region USA West (Oregon)
Südamerika	Region USA Ost (Nord-Virginia) Region USA Ost (Ohio) Region USA West (Oregon)

Erste Schritte

Um eine Zusammenfassung in natürlicher Sprache zu generieren

1. Führen Sie Ihre CloudWatch Logs Insights-Abfrage aus.
2. Wählen Sie nach Abschluss der Abfrage Ergebnisse zusammenfassen aus.

Berechtigungen

Sie müssen über eine der folgenden Voraussetzungen verfügen:

- CloudWatchLogsFullAccess-Berechtigung
- CloudWatchLogsReadOnlyAccess-Berechtigung

- Benutzerdefinierte IAM-Richtlinie, einschließlich der `cloudwatch:GenerateQueryResultsSummary` Aktionen `logs:GetQueryResults`, `logs:DescribeQueries` und `logs:FilterLogEvents`

Datenschutz

Ihre Abfrageergebnisse werden sicher verarbeitet und nicht zur Schulung oder Verbesserung von CloudWatch Logs Insights oder Amazon Bedrock verwendet. Wenn Sie sich dafür entscheiden, über die Feedback-Schaltflächen Feedback zur Zusammenfassung der Abfrageergebnisse zu geben, zeigt Ihr Feedback, wie zufrieden Sie mit den in CloudWatch Logs Insights bereitgestellten Funktionen sind.

Automatisieren der Protokollanalyse mit geplanten Abfragen

Mit geplanten Abfragen können Sie die Ausführung von CloudWatch Logs Insights-Abfragen nach einem regelmäßigen Zeitplan automatisieren. Anstatt Abfragen zur Analyse Ihrer Protokolldaten manuell auszuführen, können Sie geplante Abfragen so konfigurieren, dass sie automatisch ausgeführt werden und Ergebnisse an Ziele wie Amazon S3 S3-Buckets oder Amazon EventBridge Event Buses liefern. Diese Automatisierung ist ideal, um regelmäßige Berichte zu erstellen, Trends zu überwachen oder nachgelagerte Prozesse auf der Grundlage von Protokollanalyseergebnissen auszulösen.

Geplante Abfragen unterstützen alle drei in CloudWatch Logs Insights verfügbaren Abfragesprachen:

- [Logs Insights-Abfragesprache \(Logs Insights QL\)](#)
- [OpenSearch Sprache für die Verarbeitung per Service Piped \(PPL\)](#)
- [OpenSearch Strukturierte Abfragesprache für Dienste \(SQL\)](#)

Inhalt

- [Grundlegendes zu Konzepten für geplante Abfragen](#)
- [Referenz zum Ausdruck „Zeitplan“](#)
- [Best Practices](#)
- [Erste Schritte mit geplanten Abfragen](#)
- [Konfiguration von S3-Zielen für geplante Abfragen](#)
- [Problembehandlung bei geplanten Abfragen](#)

Grundlegendes zu Konzepten für geplante Abfragen

Bevor Sie geplante Abfragen erstellen, sollten Sie sich mit diesen Schlüsselkonzepten vertraut machen, die sich darauf auswirken, wie Ihre Abfragen ausgeführt werden und wo Ergebnisse geliefert werden.

IAM-Rollentrennung

Geplante Abfragen erfordern zwei separate IAM-Rollen: eine für die Ausführung von Abfragen und eine weitere für die Übermittlung von Ergebnissen an Amazon S3 oder Amazon EventBridge Event

Buses. Wenn Sie verstehen, warum diese Trennung existiert, können Sie Berechtigungen korrekt konfigurieren und die damit verbundenen Sicherheits- und Betriebsvorteile nutzen.

Die Architektur mit zwei Rollen verteilt die Zuständigkeiten zwischen Datenzugriff und Datenbereitstellung. Die Rolle zur Abfrageausführung greift auf Ihre Protokolldaten zu und führt Abfragen aus, während die Rolle für die Zielzustellung die Ergebnisse an das von Ihnen gewählte Ziel schreibt. Diese Trennung folgt dem Prinzip der geringsten Rechte: Jede Rolle hat nur die Berechtigungen, die sie für ihre spezifische Funktion benötigt.

Rolle zur Abfrageausführung

Ermöglicht CloudWatch Logs, CloudWatch Logs Insights-Abfragen in Ihrem Namen auszuführen. Diese Rolle benötigt Berechtigungen, um auf Ihre Protokollgruppen zuzugreifen und Abfragen auszuführen, benötigt jedoch keinen Zugriff auf Zielressourcen. Erforderliche Berechtigungen:

- `logs:StartQuery`
- `logs:StopQuery`
- `logs:GetQueryResults`
- `logs:DescribeLogGroups`
- `logs:Unmask` wenn Daten demaskiert werden müssen

Für KMS-verschlüsselte Protokollgruppen: `kms:Decrypt` und `kms:DescribeKey`
Berechtigungen für den KMS-Schlüssel, der zum Verschlüsseln der Protokollgruppen verwendet wird. Diese Berechtigungen müssen ebenfalls hinzugefügt werden.

Anforderung einer Vertrauensbeziehung: Die Rolle zur Abfrageausführung muss eine Vertrauensrichtlinie enthalten, die es dem CloudWatch Logs-Dienst (`logs.amazonaws.com`) ermöglicht, die Rolle zu übernehmen. Ohne diese Vertrauensstellung schlagen geplante Abfragen fehl und führen zu Berechtigungsfehlern.

Beispiel für eine Vertrauensrichtlinie für die Rolle zur Abfrageausführung:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.amazonaws.com"
      },
    },
  ],
}
```

```

        "Action": "sts:AssumeRole"
    }
]
}

```

Beispiel für eine Berechtigungsrichtlinie für die Rolle zur Abfrageausführung:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:StartQuery",
        "logs:StopQuery",
        "logs:GetQueryResults",
        "logs:DescribeLogGroups"
      ],
      "Resource": "*"
    }
  ]
}

```

Rolle „Zielzustellung“

Ermöglicht CloudWatch Logs, Abfrageergebnisse an das von Ihnen gewählte Ziel zu senden. Diese Rolle benötigt nur Berechtigungen für den spezifischen Zieldienst und folgt dabei dem Prinzip der geringsten Rechte. Die erforderlichen Berechtigungen variieren je nach Zieltyp.

Anforderung einer Vertrauensbeziehung: Die Zielzustellungsrolle muss auch eine Vertrauensrichtlinie enthalten, die es dem CloudWatch Logs-Dienst (`logs.amazonaws.com`) ermöglicht, die Rolle zu übernehmen.

Beispiel für eine Berechtigungsrichtlinie für die S3-Zielzustellungsrolle:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ]
    }
  ]
}

```

```
    ],  
    "Resource": "arn:aws:s3:::your-scheduled-query-results-bucket/*"  
  }  
]  
}
```

Diese Trennung bietet praktische Vorteile für Ihren Betrieb. Wenn Sie aus Sicherheitsgründen ändern müssen, wo Ergebnisse geliefert werden, ändern Sie nur die Zielzustellungsrolle, ohne die Berechtigungen zur Abfrageausführung zu ändern. Aus Gründen der Einhaltung von Vorschriften und Prüfungen können Sie eindeutig nachverfolgen, welche Rolle auf sensible Protokolldaten zugreift und welche Rolle in externe Systeme schreibt. Auf diese Weise können Sie leichter nachweisen, dass Ihre Protokollanalyse-Infrastruktur den bewährten Sicherheitsmethoden entspricht.

Regions- und kontoübergreifende Nutzung

Eine geplante Abfrage wird in einer bestimmten Region erstellt und in dieser Region ausgeführt. Sie können jedoch Protokollgruppen abfragen und Ergebnisse für alle Regionen und Konten bereitstellen. Sie müssen ein oder mehrere AWS Konten als Überwachungskonten einrichten und diese mit mehreren Quellkonten verknüpfen. Ein Überwachungskonto ist ein zentrales AWS Konto, das aus Quellkonten generierte Beobachtbarkeitsdaten einsehen und mit ihnen interagieren kann. Ein Quellkonto ist ein AWS Einzelkonto, das Beobachtbarkeitsdaten für die darin enthaltenen Ressourcen generiert. Quellkonten teilen ihre Beobachtbarkeits-Daten mit dem Überwachungskonto. Sie können also geplante Abfragen vom Überwachungskonto aus einrichten, indem Sie die Protokollgruppen aller verknüpften Konten verwenden.

Abfragen von regionsübergreifenden Protokollgruppen

Ihre geplante Abfrage kann auf Protokollgruppen in jeder Region zugreifen. Geben Sie Protokollgruppen in ihrem vollständigen ARN-Format an: `arn:aws:logs:region:account-id:log-group:log-group-name`. Die Rolle zur Abfrageausführung benötigt `logs:StartQuery` und `logs:GetQueryResults` verfügt über Berechtigungen für Protokollgruppen in allen Zielregionen.

Important

Bei der Abfrage von Protokollgruppen oder der Bereitstellung von Ergebnissen über Regionen hinweg überschreiten Protokolldaten regionale Grenzen. Berücksichtigen Sie dabei Folgendes:

- Anforderungen an den Datenstandort — Stellen Sie sicher, dass die regionsübergreifende Datenübertragung den Datenverwaltungsrichtlinien und gesetzlichen Anforderungen Ihres Unternehmens entspricht
- Kosten für die Datenübertragung — Für die regionsübergreifende Datenübertragung fallen zusätzliche Gebühren an
- Netzwerklatenz — Bei Abfragen, die auf Protokollgruppen in entfernten Regionen zugreifen, kann es zu einer höheren Latenz kommen

Für optimale Leistung und Kosteneffizienz sollten Sie geplante Abfragen in derselben Region wie Ihre primären Protokollgruppen erstellen.

Alternativer Ansatz: Verwenden Sie die [Zentralisierung von CloudWatch Logs](#), um Protokolldaten aus mehreren Konten und Regionen in ein zentrales Monitoring-Konto zu replizieren. Auf diese Weise können Sie geplante Abfragen in einer einzigen Region erstellen, die auf alle Ihre zentralen Protokolle zugreifen, wodurch regionsübergreifende Abfragen vermieden und die Verwaltung der IAM-Berechtigungen vereinfacht wird.

Zeitplanausdrücke und Zeitonenbehandlung

Der von Ihnen definierte Zeitplan bestimmt, wann Ihre Abfrage ausgeführt wird und wie oft sie ausgeführt wird. Die Auswahl des richtigen Zeitplanausdrucks wirkt sich darauf aus, wann Sie Ergebnisse erhalten und wie viele Daten Sie abfragen. Das Verständnis der Ausdruckstypen hilft Ihnen, zwischen Einfachheit und Präzision zu wählen.

Cron-Ausdrücke ermöglichen eine präzise Steuerung des Timings, sodass Sie genaue Uhrzeiten, Wochentage oder Wochentage angeben können. Verwenden Sie Cron-Ausdrücke, wenn Abfragen zu bestimmten Geschäftszeiten ausgeführt oder mit betrieblichen Zeitplänen abgestimmt werden müssen. In der Konsole können Sie Abfragen auch mithilfe einfacher Kalenderoptionen planen.

Cron-Ausdrücke

Führen Sie Abfragen zu bestimmten Zeiten aus. Format:`cron(minute hour day-of-month month day-of-week year)`. Beispiele:

- `cron(0 9 * * ? *)`- Jeden Tag um 9:00 Uhr UTC
- `cron(0 18 ? * MON-FRI *)`- Wochentags um 18:00 Uhr UTC

- `cron(0 0 1 * ? *)`- Erster Tag eines jeden Monats um Mitternacht UTC
- `cron(0 12 ? * SUN *)`- Jeden Sonntag um 12:00 Uhr UTC
- `cron(30 8 1 1 ? *)`- 1. Januar um 8:30 Uhr UTC

Alle geplanten Abfragen werden in UTC ausgeführt, unabhängig von Ihrer lokalen Zeitzone oder dem Standort Ihrer AWS Ressourcen. Dies ist besonders wichtig, wenn Sie Abfragen für Geschäftszeiten oder zeitkritische Analysen planen. Wenn Ihr Unternehmen beispielsweise in der US-Ostzeit tätig ist und Sie einen täglichen Bericht um 9 Uhr ET wünschen, müssen Sie den UTC-Offset berücksichtigen (14:00 UTC während der Sommerzeit, 13:00 UTC andernfalls). Denken Sie bei der Planung Ihrer Zeitplanausdrücke an die UTC-Zeit, um sicherzustellen, dass Abfragen zu den vorgesehenen Zeiten ausgeführt werden.

Wählen Sie eine Abfragesprache

Geplante Abfragen unterstützen drei verschiedene Abfragesprachen. Ihre Wahl wirkt sich sowohl darauf aus, wie Sie Abfragen schreiben, als auch darauf, wie einfach Ihr Team sie verwalten kann. Die richtige Sprache hängt von Ihren Analyseanforderungen und den vorhandenen Fähigkeiten Ihres Teams ab.

Wenn Sie hauptsächlich Protokolldaten filtern und aggregieren, bietet die CloudWatch Logs Insights Query Language die einfachste Syntax. Bei komplexen Datentransformationen, bei denen Sie Daten in mehreren Schritten umformen oder anreichern müssen, macht der Pipeline-Ansatz von PPL die Logik einfacher nachvollziehbar. Wenn Sie Verknüpfungen oder komplexe Aggregationen durchführen müssen, die Datenbankoperationen ähneln, bietet SQL eine vertraute Syntax, die datenbankerfahrene Teams schnell anwenden können.

CloudWatch Logs Insights Query Language (CWLI)

Speziell für die Protokollanalyse mit intuitiver Syntax entwickelt. Am besten geeignet für:

- Textbasierte Protokollanalyse und Filterung
- Aggregationen und Statistiken von Zeitreihen
- Teams, die noch keine Erfahrung mit Protokollanalyse haben

OpenSearch Service Piped Processing Language (PPL)

Pipeline-basierte Abfragesprache mit leistungsstarken Funktionen zur Datentransformation. Am besten geeignet für:

- Komplexe Datentransformationen und Anreicherung
- Mehrstufige Datenverarbeitungs-Workflows
- Teams, die mit der Verarbeitung auf Pipeline-Basis vertraut sind

OpenSearch Service Structured Query Language (SQL)

Standard-SQL-Syntax für vertraute Abfragen im Datenbankstil. Am besten geeignet für:

- Komplexe Verknüpfungen und Aggregationen
- Business Intelligence und Berichterstattung
- Teams mit langjähriger SQL-Erfahrung

Zielauswahl und Anwendungsfälle

Wohin Sie Abfrageergebnisse senden, bestimmt, was Sie mit ihnen machen können. Diese Wahl beeinflusst Ihren gesamten nachgelagerten Workflow — unabhängig davon, ob Sie Langzeitanalysen erstellen, automatisierte Antworten auslösen oder beides. Wenn Sie die Stärken der einzelnen Zieltypen kennen, können Sie die richtige Architektur für Ihren Anwendungsfall entwerfen.

Amazon S3 S3-Ziele sind für Speicherung und Stapelverarbeitung optimiert. Wenn Sie Abfrageergebnisse monatelang oder jahrelang aufbewahren, Trends im Zeitverlauf analysieren oder Daten in Analyseplattformen einspeisen müssen, bietet Amazon S3 kostengünstigen Speicherplatz mit unbegrenzter Aufbewahrung. EventBridge Ziele sind für die Automatisierung in Echtzeit optimiert. Wenn Abfrageergebnisse sofortige Aktionen auslösen sollen, z. B. das Senden von Benachrichtigungen, das Starten von Workflows oder das Aktualisieren von Systemen, werden die Ergebnisse EventBridge als Ereignisse angezeigt, auf die Ihre Anwendungen sofort reagieren können. Standardmäßig werden alle Abfrageabschlussereignisse automatisch als Ereignisse an den Standard-Event-Bus gesendet, was die Integration mit Downstream-Verarbeitungssystemen, Lambda-Funktionen oder anderen ereignisgesteuerten Architekturen ermöglicht. Ergebnisse werden nur dann an Ziele veröffentlicht, wenn die Abfrage erfolgreich ausgeführt wurde.

Amazon-S3-Ziele

Speichern Sie Abfrageergebnisse als JSON-Dateien für die langfristige Aufbewahrung und Stapelverarbeitung. Am besten geeignet für:

- Historische Analyse und Datenarchivierung
- Integration mit Data Lakes und Analyseplattformen
- Konformitäts- und Prüfanforderungen

- Kostengünstige Speicherung großer Ergebnismengen

EventBridge Destinationen

Senden Sie Abfrageergebnisse als Ereignisse für die Verarbeitung und Automatisierung in Echtzeit. Sie können Abfrageergebnisse mit der im Ereignis gesendeten `queryId` nur bis zu 30 Tage abrufen, da wir die Ergebnisse für 30 Tage speichern. Am besten geeignet für:

- Auslösen automatisierter Antworten auf Abfrageergebnisse
- Integration mit serverlosen Workflows und Lambda-Funktionen
- Warn- und Benachrichtigungssysteme in Echtzeit
- Ereignisgesteuerte Architekturen und Microservices

Format und Struktur der Abfrageergebnisse

Für Amazon S3 S3-Ziele — Abfrageergebnisse werden im JSON-Format mit derselben Struktur wie die `GetQueryResults` API-Antwort geliefert. Für Amazon hilft Ihnen das EventBridge Verständnis des Formats der geplanten Abfrageergebnisse bei der Gestaltung nachgelagerter Verarbeitungs- und Integrationsworkflows.

Abfrageergebnisse werden im JSON-Format mit der folgenden Struktur geliefert:

```
{
  "version": "0",
  "id": "be72061b-eca2-e068-a7e1-83e01d6fe807",
  "detail-type": "Scheduled Query Completed",
  "source": "aws.logs",
  "account": "123456789012",
  "time": "2025-11-18T11:31:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:logs:us-east-1:123456789012:scheduled-query:477b4380-b098-474e-9c5e-e10a8cc2e6e7"
  ],
  "detail": {
    "queryId": "2038fd57-ab4f-4018-bb2f-61d363f4a004",
    "queryString": "fields @timestamp, @message, @logStream, @log\n| sort\n@timestamp desc\n| limit 10000",
    "logGroupIdentifiers": [
      "/aws/lambda/my-function"
    ],
  },
}
```

```

    "status": "Complete",
    "startTime": 1763465460,
    "statistics": {
      "recordsMatched": 0,
      "recordsScanned": 0,
      "estimatedRecordsSkipped": 0,
      "bytesScanned": 0,
      "estimatedBytesSkipped": 0,
      "logGroupsScanned": 1
    }
  }
}

```

Zu den wichtigsten Elementen gehören:

- **statistics**- Leistungskennzahlen für Abfragen, einschließlich übereinstimmender, gescannter, verarbeiteter Byte und geschätzter übersprungener Daten
- **startTime**— Wann die Ausführung der Abfrage gestartet wurde (Unix-Zeitstempel)
- **queryString**- Die eigentliche Abfrage, die ausgeführt wurde
- **queryId**- Abfrage-ID der Abfrage, mit der Ergebnisse abgerufen werden können
- **logGroupIdentifiers**- Liste der Protokollgruppen, die abgefragt wurden
- **status**- Ausführungsstatus abfragen (Abgeschlossen, Fehlgeschlagen usw.)

Referenz zum Ausdruck „Zeitplan“

Verwenden Sie diese Referenztabelle, um Zeitplanausdrücke für Ihre geplanten Abfragen zu erstellen. Alle Uhrzeiten sind in UTC angegeben.

Syntax von Cron-Ausdrücken

Format: `cron(minute hour day-of-month month day-of-week year)`

Anwendungsfall	Cron-Ausdruck	Description	Verwenden Sie Wann
Tagespläne	<code>cron(0 9 * * ? *)</code>	Jeden Tag um 9:00 Uhr UTC	Tägliche Berichte

Anwendungsfall	Cron-Ausdruck	Description	Verwenden Sie Wann
	<code>cron(0 */6 * * ? *)</code>	Alle 6 Stunden (00:00, 06:00, 12:00, 18:00 UTC)	Häufige Überwachung
	<code>cron(30 2 * * ? *)</code>	Jeden Tag um 2:30 Uhr UTC	Analyse außerhalb der Spitzenzeiten
Geschäftszeiten	<code>cron(0 9-17 ? * MON-FRI *)</code>	Jede Stunde von 9 Uhr bis 17 Uhr, Montag bis Freitag UTC	Überwachung von Unternehmen
	<code>cron(0 18 ? * MON-FRI *)</code>	Wochentags um 18:00 Uhr UTC	Ende des Geschäftstages
	<code>cron(0 8,12,17 ? * MON-FRI *)</code>	8 Uhr, 12 Uhr und 17 Uhr an Wochentagen UTC	Wichtige Geschäftszeiten
Wöchentliche Zeitpläne	<code>cron(0 12 ? * SUN *)</code>	Jeden Sonntag um 12 Uhr UTC	Wöchentliche Zusammenfassungen
	<code>cron(0 9 ? * MON *)</code>	Jeden Montag um 9:00 Uhr UTC	Berichte zum Wochenstart

Anwendungsfall	Cron-Ausdruck	Description	Verwenden Sie Wann
	<code>cron(0 23 ? * FRI *)</code>	Jeden Freitag um 23:00 Uhr UTC	Aufräumen am Wochenende
Monatliche Zeitpläne	<code>cron(0 0 1 * ? *)</code>	Erster Tag eines jeden Monats um Mitternacht UTC	Monatliche Berichte
	<code>cron(0 9 L * ? *)</code>	Letzter Tag jedes Monats um 9:00 Uhr UTC	Verarbeitung zum Monatsende
	<code>cron(0 10 1 1,4,7,10 ? *)</code>	Erster Tag jedes Quartals um 10:00 Uhr UTC	Vierteljährliche Analyse
Hohe Frequenz	<code>cron(* /15 * * * ? *)</code>	Alle 15 Minuten	Überwachung in Echtzeit
	<code>cron(0,30 * * * ? *)</code>	Alle 30 Minuten (um :00 Uhr und :30 Uhr)	Häufige Kontrollen
	<code>cron(0 */2 * * ? *)</code>	Alle 2 Stunden	Regelmäßige Intervalle
Sonderfälle	<code>cron(30 8 1 1 ? *)</code>	1. Januar um 8:30 Uhr UTC	Jahresberichte
	<code>cron(0 6 * * SAT,SUN *)</code>	Am Wochenende um 6:00 Uhr UTC	Bearbeitung am Wochenende

Anwendungsfall	Cron-Ausdruck	Description	Verwenden Sie Wann
	<code>cron(0 0 ? * MON#1 *)</code>	Jeden ersten Montag im Monat um Mitternacht UTC	Monatliche Planung

Feldverweis auf den Cron-Ausdruck

Feld	Werte	Platzhalter	Beispiele
Minute (1.)	0-59	* , - /	0(Höhepunkt der Stunde), */15 (alle 15 Minuten), 0, 30 (zweimal pro Stunde)
Stunde (2.)	0-23	* , - /	9(9 Uhr), */2 (alle 2 Stunden), 9-17 (Geschäftszeiten)
Day-of-month (3.)	1-31, L, W	* , - / ?	1(1. Tag), L (letzter Tag), ? (bei Verwendung day-of-week)
Monat (4.)	1-12 oder JAN-DEC	* , - /	1(Januar)JAN, 1, 4, 7, 10 (vierteljährlich)
Day-of-week (5.)	1-7 oder SUN-SAT	* , - / ? # L	MON-FRI(wochentags)SUN,, MON#1 (1. Montag)
Jahr (6.)	1970-2199	* , - /	*(jedes Jahr), 2024 (bestimmtes Jahr), 2024-2026 (Bereich)

Platzhalterzeichen und Sonderausdrücke

* (Sternchen)

Entspricht allen Werten im Feld. Beispiel: * Im Stundenfeld bedeutet das jede Stunde.

?(Fragezeichen)

Kein spezifischer Wert. Wird in day-of-month oder verwendet day-of-week, wenn der andere Wert angegeben ist. Beispiel: Verwenden Sie ? in day-of-month, wenn Sie MON-FRI in angeben day-of-week.

- (Bindestrich)

Wertebereich. Beispiel: MON-FRI (Montag bis Freitag), 9-17 (9 Uhr bis 17 Uhr).

,(Komma)

Mehrere spezifische Werte. Beispiel: MON, WED, FRI (Montag, Mittwoch, Freitag), 8, 12, 17 (8 Uhr, Mittag, 17 Uhr).

/ (Schrägstrich)

Schrittweite oder Inkremente. Beispiel: 0/15 In Minuten bedeutet alle 15 Minuten ab Minute 0 (0, 15, 30, 45). */2 in Stunden bedeutet alle 2 Stunden.

L(zuletzt)

Letzter Tag des Monats oder letztes Vorkommen des Wochentags. Beispiel: L In day-of-month bedeutet letzter Tag des Monats. FRIL bedeutet letzten Freitag im Monat.

W(Wochentag)

Nächster Wochentag. Beispiel: 15W bedeutet, dass der Wochentag dem 15. des Monats am nächsten liegt.

#(n-tes Vorkommen)

N-tes Vorkommen eines Wochentags in einem Monat. Beispiel: MON#1 bedeutet ersten Montag im Monat, FRI#2 bedeutet zweiten Freitag im Monat.

Allgemeine Muster und bewährte Verfahren

- Für Geschäftsanwendungen: Nutzung MON-FRI und Geschäftszeiten (z. B. 9-17), um zu vermeiden, dass Anfragen an Wochenenden oder außerhalb der Geschäftszeiten laufen.
- Für die Überwachung mit hoher Frequenz: Verwenden Sie Inkremente wie */15 (alle 15 Minuten), beachten Sie jedoch die Grenzwerte für die Parallelität von Abfragen.
- Aus Gründen der Ressourceneffizienz: Planen Sie ressourcenintensive Abfragen außerhalb der Spitzenzeiten und verwenden Sie dabei frühe Morgenzeiten wie UTC. 2-6

- Für monatliche Berichte: Verwenden Sie diese Option L für den letzten Tag des Monats oder für bestimmte Daten, z. B. 1 für den ersten Tag, um ein einheitliches Timing zu gewährleisten.

Best Practices

Halten Sie sich an die folgenden bewährten Methoden, um zuverlässige und effiziente geplante Abfragevorgänge sicherzustellen:

Optimierung von Abfragen

- Testen Sie Abfragen vor der Planung manuell, um Leistung und Ergebnisse zu überprüfen
- Verwenden Sie zu Beginn Ihrer Abfrage Filterindizes, um die Datenverarbeitung zu reduzieren
- Begrenzen Sie Zeitbereiche, um Timeouts bei Protokollgruppen mit hohem Volumen zu vermeiden
- Berücksichtigen Sie die Komplexität der Abfragen und die Zeitlimits für die Ausführung

Planung des Zeitplans

- Vermeiden Sie sich überschneidende Ausführungen, indem Sie sicherstellen, dass Abfragen vor der nächsten geplanten Ausführung abgeschlossen sind
- Berücksichtigen Sie bei der Festlegung von Zeitbereichen Verzögerungen bei der Protokollaufnahme
- Verwenden Sie Cron-Ausdrücke für bestimmte Zeiten
- Verteilen Sie die Zeitpläne, um sicherzustellen, dass Sie das Limit für die Parallelität von Abfragen nicht überschreiten

Überwachung und Wartung

- Überwachen Sie den Ausführungsverlauf regelmäßig, um Ausfälle oder Leistungsprobleme zu identifizieren
- Überprüfen und aktualisieren Sie die IAM-Rollen regelmäßig, um die Sicherheit zu gewährleisten
- Testen Sie den Zugriff auf das Ziel, bevor Sie es in der Produktion einsetzen

Autorisierung

- Alle Abfragen APIs für geplante Abfragen autorisieren für die geplante Abfrageressource und nicht für die Ressourcen, die für die Eingabe benötigt werden, wie z. B. Protokollgruppen. Richten Sie die IAM-Richtlinien entsprechend ein

- Verwalten Sie die Autorisierung von Protokollgruppen mithilfe der Ausführungsrolle, die in der APIs

Erste Schritte mit geplanten Abfragen

Wenn Sie eine geplante Abfrage erstellen, konfigurieren Sie mehrere wichtige Komponenten, die definieren, wie Ihre Abfrage ausgeführt wird und wohin die Ergebnisse geliefert werden. Wenn Sie sich mit diesen Komponenten vertraut machen, können Sie eine effektive automatisierte Protokollanalyse einrichten.

Jede geplante Abfrage besteht aus den folgenden Schlüsselkomponenten:

Konfiguration der Abfrage

Die CloudWatch Logs Insights-Abfragezeichenfolge, die Zielprotokollgruppen und die Abfragesprache, die für die Analyse verwendet werden sollen.

Ausdruck zeitlich planen

Ein Cron-Ausdruck oder ein Frequenzkalender, der definiert, wann die Abfrage ausgeführt wird. Sie können Zeitzoneneinstellungen angeben, um sicherzustellen, dass Abfragen zur richtigen Ortszeit ausgeführt werden. Die Konsole zeigt eine für Menschen lesbare Beschreibung Ihres Zeitplans an, z. B. „Führen Sie die Abfrage jeden Dienstag um 15:10 Uhr für einen Zeitraum von 5 Minuten aus, mit sofortiger Wirkung auf UTC, bis auf unbestimmte Zeit“.

Zeitraum

Der Lookback-Zeitraum für jede Abfrageausführung, definiert durch einen Abstand zwischen der Startzeit und der Ausführungszeit. Dies bestimmt, wie viele historische Daten bei jeder Abfrageausführung analysiert werden.

Vorschau des Ausführungsplans

Die Konsole zeigt die nächsten drei geplanten Abfrageläufe mit genauen Datums- und Uhrzeitangaben an (z. B. 2025/10/28 15:10, UTC; 2025/11/04 15:10, UTC; 2025/11/11 15:10, UTC), sodass Sie überprüfen können, ob der Zeitplan korrekt konfiguriert ist.

Ziele

Wo Abfrageergebnisse nach erfolgreicher Ausführung geliefert werden. Zu den unterstützten Zielen gehören Amazon S3 S3-Buckets, und standardmäßig werden Ergebnismetadaten an den Standard-Event-Bus gesendet.

Rolle „Ausführung“

Eine IAM-Rolle, von der CloudWatch Logs annimmt, um die Abfrage auszuführen und Ergebnisse an die angegebenen Ziele zu liefern.

Stellen Sie vor dem Erstellen von geplanten Abfragen sicher, dass Sie die erforderlichen Berechtigungen und Ressourcen konfiguriert haben.

Eine geplante Abfrage erstellen

Erstellen Sie eine geplante Abfrage, die automatisch CloudWatch Logs Insights-Abfragen ausführt und Ergebnisse an die von Ihnen ausgewählten Ziele liefert.

Voraussetzungen

Bevor Sie eine geplante Abfrage erstellen, stellen Sie sicher, dass Sie über Folgendes verfügen:

- Protokollgruppen — Eine oder mehrere Protokollgruppen, die die Daten enthalten, die Sie analysieren möchten
- Ausführungs-IAM-Rolle — Eine IAM-Rolle mit den folgenden Berechtigungen:
 - `logs:StartQuery`— Berechtigung zum Starten von CloudWatch Logs Insights-Abfragen
 - `logs:GetQueryResults`- Erlaubnis zum Abrufen von Abfrageergebnissen
 - `logs:DescribeLogGroups`- Berechtigung zum Zugriff auf Protokollgruppeninformationen. Dies ist nur für präfixbasierte Protokollgruppen zur Erkennung von Protokollgruppen erforderlich
- Zielberechtigungen — Zusätzliche IAM-Berechtigungen für das von Ihnen gewählte Ziel:
 - Für Amazon S3 S3-Ziele: `s3:PutObject`
- Für AWS CLI und API-Nutzung — Konfigurierte AWS Anmeldeinformationen mit Berechtigungen zum Aufrufen von CloudWatch Logs APIs

Ausführliche Beispiele für IAM-Richtlinien finden Sie unter [Identitäts- und Zugriffsmanagement für Amazon CloudWatch Logs](#). Beachten Sie außerdem, dass Sie pro Konto nur 1000 geplante Abfragen haben können.

Console

Um eine geplante Abfrage zu erstellen (Konsole)

1. Die CloudWatch Logs-Konsole zu [https://us-east-1.console.aws.amazon.com/cloudwatch/Hause öffnen? region=us-east-1 #logsV2: Logs-Einblicke](https://us-east-1.console.aws.amazon.com/cloudwatch/Hause%20öffnen?region=us-east-1#logsV2:Logs-Einblicke).
2. Wählen Sie im Navigationsbereich Logs Insights aus.
3. Wählen Sie Geplante Abfrage erstellen aus.
4. Gehen Sie im Abschnitt Abfragedefinition wie folgt vor:
 - a. Wählen Sie unter Abfragesprache die zu verwendende Abfragesprache aus der Liste aus.
 - b. Geben Sie als Abfragezeichenfolge Ihre CloudWatch Logs Insights-Abfrage in das Feld ein.
 - c. Wählen Sie für Protokollgruppen die abzufragenden Protokollgruppen aus der Liste aus.
5. Gehen Sie im Abschnitt Einrichtung des Zeitplans wie folgt vor:
 - a. Konfigurieren Sie für den Ausdruck Schedule, wann die Abfrage ausgeführt wird. Wählen Sie aus vordefinierten Optionen oder geben Sie einen benutzerdefinierten Cron-Ausdruck ein.
 - b. Geben Sie unter Gültig bei der Erstellung an, wann der Zeitplan aktiv wird. Wählen Sie mithilfe des YYYY/MM/DD Formats, ob der Start sofort oder an einem bestimmten Datum und zu einer bestimmten Uhrzeit erfolgen soll.
 - c. Geben Sie unter Zeitraum den Lookback-Zeitraum für jede Abfrageausführung an. Geben Sie die Dauer in Minuten ein, die definiert, wie weit die Abfrage von der Ausführungszeit zurückliegt.
 - d. Geben Sie für Auf unbestimmte Zeit fortsetzen an, wann der Zeitplan endet. Wählen Sie mithilfe des Formats, ob die Ausführung auf unbestimmte Zeit oder bis zu einem bestimmten Datum und einer bestimmten Uhrzeit erfolgen YYYY/MM/DD soll.
6. In der Konsole werden die nächsten drei geplanten Abfrageläufe basierend auf Ihrer Konfiguration angezeigt. Dabei werden die genauen Datums- und Uhrzeitangaben in UTC angezeigt, zu denen die Abfrage ausgeführt wird.
7. Im Abschnitt Abfrageergebnisse an S3 senden — optional (wenn Sie das S3-Ziel verwenden):
 - a. Wählen Sie für S3-Bucket Dieses Konto aus, wenn sich der Ziel-Bucket in demselben AWS Konto befindet, oder wählen Sie Anderes Konto aus, wenn sich der Bucket in

- einem anderen AWS Konto befindet, und geben Sie die Konto-ID des Kontos, das den Bucket besitzt, als Eingabe ein.
- b. Geben Sie für Amazon S3 S3-URI den Amazon S3 S3-Bucket und das Präfix ein, in dem die Ergebnisse gespeichert werden (z. B. `s3://my-bucket/query-results/`). Wenn Sie Dieses Konto ausgewählt haben, können Sie Amazon S3 durchsuchen wählen, um zu einem vorhandenen Amazon S3 S3-Standort zu navigieren und diesen auszuwählen.
 - c. (Optional) Geben Sie für den KMS-Schlüssel-ARN den ARN eines vom Kunden verwalteten AWS KMS Schlüssels ein, um die Abfrageergebnisse mit SSE-KMS zu verschlüsseln. Der Schlüssel muss sich in derselben AWS Region wie der Amazon S3-Ziel-Bucket befinden.
8. Wählen Sie im Abschnitt IAM-Rolle für die Veröffentlichung von Abfrageergebnissen in Amazon S3 eine der folgenden Optionen aus:
- a. Wählen Sie Neue Rolle mit Standardberechtigungen automatisch erstellen, um automatisch eine IAM-Rolle mit den erforderlichen Berechtigungen einzurichten, damit CloudWatch Logs Abfrageergebnisse an Amazon S3 liefern kann.
 - b. Wählen Sie Bestehende Rolle verwenden, um eine bestehende IAM-Rolle mit den erforderlichen Richtlinien für CloudWatch Logs auszuwählen, um Abfrageergebnisse an Amazon S3 zu liefern. Verwenden Sie das Suchfeld, um die entsprechende IAM-Rolle aus der Liste zu finden und auszuwählen.
9. Wählen Sie im Abschnitt IAM-Rolle für die geplante Abfrageausführung eine der folgenden Optionen aus:
- a. Wählen Sie Neue Rolle mit Standardberechtigungen automatisch erstellen, um automatisch eine IAM-Rolle mit den für CloudWatch Logs erforderlichen Berechtigungen einzurichten, um geplante Abfragen auszuführen.
 - b. Wählen Sie „Bestehende Rolle verwenden“, um eine bestehende IAM-Rolle mit den erforderlichen Richtlinien für CloudWatch Logs auszuwählen, um geplante Abfragen auszuführen. Verwenden Sie das Suchfeld, um die entsprechende IAM-Rolle aus der Liste zu finden und auszuwählen.
10. Wählen Sie Zeitplan erstellen, um die geplante Abfrage zu erstellen.

AWS CLI

Um eine geplante Abfrage zu erstellen (AWS CLI)

- Verwenden Sie den `create-scheduled-query` Befehl, um eine neue geplante Abfrage zu erstellen:

```
aws logs create-scheduled-query \
  --name "ErrorAnalysisQuery" \
  --query-language "CWL" \
  --query-string "fields @timestamp, @message | filter @message like /ERROR/ |
stats count() by bin(5m)" \
  --schedule-expression "cron(8 * * * ? *)" \
  --execution-role-arn "arn:aws:iam::123456789012:role/
CloudWatchLogsScheduledQueryRole" \
  --log-group-identifiers "/aws/lambda/my-function" "/aws/apigateway/my-api" \
  --state "ENABLED"
```

API

Um eine geplante Abfrage (API) zu erstellen

- Verwenden Sie die `CreateScheduledQuery` Aktion, um eine neue geplante Abfrage zu erstellen. Im folgenden Beispiel wird eine geplante Abfrage erstellt, die jede Stunde ausgeführt wird:

```
{
  "name": "ErrorAnalysisQuery",
  "queryLanguage": "CWL",
  "queryString": "fields @timestamp, @message | filter @message like /ERROR/ |
stats count() by bin(5m)",
  "scheduleExpression": "cron(8 * * * ? *)",
  "executionRoleArn": "arn:aws:iam::123456789012:role/
CloudWatchLogsScheduledQueryRole",
  "logGroupIdentifiers": ["/aws/lambda/my-function", "/aws/apigateway/my-
api"],
  "state": "ENABLED"
}
```

Nachdem Sie die geplante Abfrage erstellt haben, können Sie sie auf der Seite Geplante Abfragen und mithilfe der ListScheduledQueries API anzeigen und verwalten. Dort werden alle Ihre geplanten Abfragen mit ihren Namen, Erstellungsdaten, dem Status der letzten Ausführung, der Uhrzeit der letzten Auslösung und der Wiederholungshäufigkeit angezeigt.

Geplante Abfragen anzeigen und verwalten

Die folgenden Informationen sind für jede Abfrage verfügbar:

Name

Der eindeutige Name, den Sie der geplanten Abfrage zugewiesen haben. Wählen Sie den Namen aus, um den detaillierten Verlauf der Konfiguration und Ausführung anzuzeigen.

Erstellungsdatum

Das Datum, an dem die geplante Abfrage erstellt wurde, wird im YYYY-MM-DD Format angezeigt.

Status der letzten Ausführung

Der Ausführungsstatus des letzten Abfragelaufs. Mögliche Werte sind:

- Abgeschlossen — Die Abfrage wurde erfolgreich ausgeführt und die Ergebnisse wurden an alle konfigurierten Ziele übermittelt.
- Fehlgeschlagen — Die Ausführung der Abfrage oder die Übermittlung der Ergebnisse ist fehlgeschlagen. Einzelheiten zu den Fehlern finden Sie im Ausführungsverlauf.
- Ungültige Abfrage — Die Abfrage ist ungültig und weist syntaktische Probleme auf
- Timeout — Das Zeitlimit für die Abfrage wurde überschritten. Bei einer Abfrage wird das Timeout automatisch nach 60 Minuten überschritten

Zeitpunkt des letzten Auslösens

Datum und Uhrzeit der letzten Ausführung der Abfrage, angezeigt im YYYY-MM-DD Format HH:MM:SS. Zeigt Nie an, wenn die Abfrage noch nicht ausgeführt wurde.

Alle werden wiederholt

Die Häufigkeit des Zeitplans für die Abfrage. Zeigt Benutzerdefiniert für Abfragen an, die Cron-Ausdrücke oder spezifische Häufigkeitsbeschreibungen für einfachere Zeitpläne verwenden.

Die Seite Geplante Abfragen bietet einen Überblick über all Ihre geplanten Abfragen und zeigt deren aktuellen Status und Ausführungsverlauf an, sodass Sie all Ihre geplanten Abfragen von einem

zentralen Ort aus anzeigen, überwachen und verwalten können. Verwenden Sie diese Informationen, um die Abfrageleistung zu überwachen, Probleme zu identifizieren und Ihre automatisierten Protokollanalyse-Workflows zu verwalten.

Console

So zeigen Sie geplante Abfragen an (Konsole)

1. Die CloudWatch Logs-Konsole zu [https://us-east-1.console.aws.amazon.com/cloudwatch/Hause öffnen? region=us-east-1 #logsV2: Logs-Einblicke](https://us-east-1.console.aws.amazon.com/cloudwatch/Hause%20öffnen?region=us-east-1#logsV2:Logs-Einblicke).
2. Wählen Sie in der CloudWatch Logs-Konsole die Optionen Geplante Abfrage und Geplante Abfragen anzeigen aus.

AWS CLI

Um geplante Abfragen aufzulisten (AWS CLI)

- Verwenden Sie den `list-scheduled-queries` Befehl, um alle geplanten Abfragen aufzulisten:

```
aws logs list-scheduled-queries --max-results 10
```

API

Um geplante Abfragen aufzulisten (API)

- Verwenden Sie die `ListScheduledQueries` Aktion, um alle geplanten Abfragen abzurufen:

```
{
  "maxResults": 10
}
```

In der Kopfzeile der Seite Geplante Abfragen wird die Gesamtzahl der geplanten Abfragen in Ihrem Konto angezeigt, sodass Sie Ihre Nutzung verfolgen und Ihre automatisierten Protokollanalyse-Workflows effektiv verwalten können.

Den Verlauf der Ausführung geplanter Abfragen anzeigen

Verwenden Sie den Ausführungsverlauf, um die Leistung Ihrer geplanten Abfragen zu überwachen und Probleme bei der Abfrageausführung oder Ergebnislieferung zu beheben.

In der Ausführungshistorie wird der Status der einzelnen Abfrageläufe angezeigt, einschließlich erfolgreicher Ausführungen, Fehlschläge und Ergebnisse der Zielverarbeitung. Sie können diese Informationen verwenden, um Muster zu identifizieren, Probleme zu diagnostizieren und zu überprüfen, ob Ihre Abfragen erwartungsgemäß ausgeführt werden.

Console

So zeigen Sie den Ausführungsverlauf an (Konsole)

1. Wählen Sie in der CloudWatch Protokollkonsole die Optionen Geplante Abfrage und Geplante Abfragen anzeigen aus.
2. Wählen Sie die geplante Abfrage aus, die Sie untersuchen möchten.
3. Wählen Sie die Registerkarte Execution history (Ausführungsverlauf).

AWS CLI

Um den Ausführungsverlauf anzuzeigen (AWS CLI)

1. Verwenden Sie den `get-scheduled-query-history` Befehl, um den Ausführungsverlauf für eine geplante Abfrage abzurufen:

```
aws logs get-scheduled-query-history \
  --identifier "DailyErrorMonitoring" \
  --start-time 1743379200 \
  --end-time 1743465600 \
  --max-results 10
```

2. Um nach dem Ausführungsstatus zu filtern, fügen Sie den folgenden `--execution-statuses` Parameter hinzu:

```
aws logs get-scheduled-query-history \
  --identifier "DailyErrorMonitoring" \
  --start-time 1743379200 \
  --end-time 1743465600 \
  --max-results 1 \
  --execution-statuses
```



```
--execution-statuses "SUCCEEDED"
```

API

Um den Ausführungsverlauf (API) anzuzeigen

- Verwenden Sie die `GetScheduledQueryHistory` Aktion, um den Ausführungsverlauf abzurufen:

```
{
  "identifizier": "DailyErrorMonitoring",
  "startTime": 1743379200,
  "endTime": 1743465600,
  "maxResults": 10,
  "executionStatuses": ["SUCCEEDED", "FAILED"]
}
```

In der Ausführungshistorie wird Folgendes angezeigt:

- Ausführungsstatus — Wird ausgeführt, Abgeschlossen, Fehlgeschlagen, Timeout oder `InvalidQuery`
- Ausgelöste Zeit — Wann die Abfrage ausgeführt wurde
- Ziele — Verarbeitungsstatus für jedes konfigurierte Ziel, einschließlich S3 und EventBridge
- Fehlermeldungen — Einzelheiten zu Fehlern bei der Abfrageausführung oder Zielverarbeitung

Aktualisierung einer geplanten Abfrage

Ändern Sie Ihre Konfiguration für geplante Abfragen, um die Abfragezeichenfolge, den Zeitplan, die Ziele oder die Ausführungsrolle entsprechend Ihren Anforderungen zu ändern.

Sie können jeden Aspekt einer geplanten Abfrage aktualisieren, einschließlich der Abfragezeichenfolge, des Zeitplanausdrucks, der Ziele und der Ausführungsrolle. Änderungen werden sofort für future Ausführungen wirksam.

Console

Um eine geplante Abfrage zu aktualisieren (Konsole)

1. Wählen Sie in der CloudWatch Protokollkonsole die Optionen Geplante Abfrage und Geplante Abfragen anzeigen aus.
2. Wählen Sie die geplante Abfrage aus, die Sie aktualisieren möchten.
3. Wählen Sie Bearbeiten aus.
4. Ändern Sie die Konfiguration nach Bedarf.
5. Wählen Sie Änderungen speichern aus.

AWS CLI

Um eine geplante Abfrage zu aktualisieren (AWS CLI)

- Verwenden Sie den `update-scheduled-query` Befehl, um eine bestehende geplante Abfrage zu ändern:

```
aws logs update-scheduled-query \
  --identifier "arn:aws:logs:us-east-1:111122223333:scheduled-
query:5e0c0228-1c29-4d26-904f-59f1f1ba3c8f" \
  --description "Monitor for ERROR level logs daily" \
  --query-language "LogsQL" \
  --query-string "fields @timestamp, @message | filter @message like /ERROR/"
\
  --log-group-identifiers "/aws/lambda/my-function-1" "/aws/lambda/my-
function-2"
```

API

Um eine geplante Abfrage (API) zu aktualisieren

1. Verwenden Sie die `UpdateScheduledQuery` Aktion, um die Konfiguration der geplanten Abfrage zu ändern:

```
{
  "identifier": "arn:aws:logs:us-east-1:111122223333:scheduled-
query:5e0c0228-1c29-4d26-904f-59f1f1ba3c8f",
```

```

    "queryString": "fields @timestamp, @message | filter @message like /WARNING|
    ERROR/ | stats count() by bin(5m)",
    "scheduleExpression": "cron(0 */2 * * ? *)",
    "state": "ENABLED"
  }

```

2. Um mehrere Konfigurationsparameter gleichzeitig zu aktualisieren:

```

{
  "identifizier": "arn:aws:logs:us-east-1:111122223333:scheduled-
  query:5e0c0228-1c29-4d26-904f-59f1f1ba3c8f",
  "queryString": "fields @timestamp, @message, @level | filter @level =
  'ERROR'",
  "scheduleExpression": "cron(0 8,12,16 * * ? *)",
  "executionRoleArn": "arn:aws:iam::111122223333:role/
  UpdatedScheduledQueryRole",
  "logGroupIdentifiers": ["/aws/lambda/my-function", "/aws/lambda/another-
  function"],
  "destinationConfiguration": {
    "s3Configuration": {
      "destinationIdentifier": "s3://111122223333-sqn-results-bucket/
      processed-results",
      "roleArn": "arn:aws:iam::111122223333:role/Admin"
    }
  }
}

```

Konfiguration von S3-Zielen für geplante Abfragen

Konfigurieren Sie Amazon S3 als Ziel, um Ihre geplanten Abfrageergebnisse als JSON-Dateien zur langfristigen Aufbewahrung und Analyse zu speichern.

Wenn Sie Amazon S3 als Ziel verwenden, werden Abfrageergebnisse als JSON-Dateien im angegebenen Bucket und Präfix gespeichert. Diese Option ist ideal für die Archivierung von Ergebnissen, die Durchführung von Batch-Analysen oder die Integration mit anderen AWS Diensten, die S3-Daten verarbeiten.

Sie können Abfrageergebnisse an einen Amazon S3 S3-Bucket in demselben AWS Konto wie die geplante Abfrage oder an einen Bucket in einem anderen AWS Konto senden. Optional können Sie Abfrageergebnisse auch mit einem vom Kunden verwalteten AWS KMS Schlüssel (SSE-KMS) verschlüsseln.

Ergebnisse an einen Amazon S3 S3-Bucket im selben Konto senden

Wenn sich der Amazon S3 S3-Ziel-Bucket in demselben AWS Konto wie die geplante Abfrage befindet, können Sie den Bucket direkt von der Konsole aus suchen und auswählen.

So konfigurieren Sie ein Amazon S3 S3-Ziel mit demselben Konto (Konsole)

1. Wählen Sie im Abschnitt Abfrageergebnisse an S3 senden für S3-Bucket die Option Dieses Konto aus.
2. Geben Sie für Amazon S3-URI den Amazon S3 S3-Bucket und das Präfix ein, in dem die Ergebnisse gespeichert werden (z. B. `s3://my-bucket/query-results/`), oder wählen Sie Amazon S3 durchsuchen, um zu navigieren und einen vorhandenen Amazon S3 S3-Standort auszuwählen.
3. (Optional) Um Ergebnisse mit einem vom Kunden verwalteten AWS KMS Schlüssel zu verschlüsseln, geben Sie den ARN des AWS KMS Schlüssels in das Feld KMS-Schlüssel-ARN ein. Der Schlüssel muss sich in derselben AWS Region wie der Amazon S3-Ziel-Bucket befinden. Wenn Sie keinen AWS KMS Schlüssel angeben, gelten die Standard-Verschlüsselungseinstellungen des Buckets.
4. Wählen Sie im Abschnitt IAM-Rolle für die Veröffentlichung von Abfrageergebnissen in Amazon S3 die Option Neue Rolle mit Standardberechtigungen automatisch erstellen aus, um die erforderlichen Berechtigungen automatisch einzurichten, oder wählen Sie Bestehende Rolle verwenden, um eine bestehende IAM-Rolle mit den erforderlichen Richtlinien auszuwählen.

Die IAM-Rolle für die Zielzustellung erfordert die folgenden Berechtigungen:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3::my-bucket/prefix/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

}

Ergebnisse an einen Amazon S3 S3-Bucket in einem anderen Konto senden

Sie können geplante Abfrageergebnisse an einen Amazon S3 S3-Bucket in einem anderen AWS Konto senden. Wenn Sie einen kontoübergreifenden Bucket verwenden, müssen Sie den Amazon S3 S3-URI und die Konto-ID des Kontos angeben, das den Bucket besitzt.

So konfigurieren Sie ein kontoübergreifendes Amazon S3 S3-Ziel (Konsole)

1. Wählen Sie im Abschnitt Abfrageergebnisse an S3 senden für S3-Bucket die Option Anderes Konto aus und geben Sie die Konto-ID des Kontos, das den Bucket besitzt, als Eingabe ein.
2. Geben Sie für Amazon S3 S3-URI die vollständige Amazon S3 S3-URI des Ziel-Buckets und das Präfix im anderen Konto ein (z. B. `s3://cross-account-bucket/query-results/`).
3. (Optional) Um Ergebnisse mit einem vom Kunden verwalteten AWS KMS Schlüssel zu verschlüsseln, geben Sie den ARN des AWS KMS Schlüssels in das Feld KMS-Schlüssel-ARN ein. Der Schlüssel muss sich in derselben AWS Region wie der Amazon S3-Ziel-Bucket befinden.
4. Wählen Sie im Abschnitt IAM-Rolle für die Veröffentlichung von Abfrageergebnissen in Amazon S3 die Option Neue Rolle mit Standardberechtigungen automatisch erstellen aus, um die erforderlichen Berechtigungen automatisch einzurichten, oder wählen Sie Bestehende Rolle verwenden, um eine bestehende IAM-Rolle mit den erforderlichen Richtlinien auszuwählen.

Für die kontoübergreifende Lieferung sind Genehmigungen auf beiden Seiten erforderlich. Die IAM-Rolle für die Zielzustellung im Quellkonto erfordert die folgenden Berechtigungen:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::cross-account-bucket/prefix/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

```

    }
  }
]
}

```

Die Amazon S3 S3-Bucket-Richtlinie im Zielkonto muss der IAM-Rollenberechtigung des Quellkontos die Berechtigung zum Schreiben von Objekten gewähren:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowScheduledQueryRolePutObject",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/my-s3-delivery-role"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3::cross-account-bucket/prefix/*"
    }
  ]
}

```

Verschlüsseln von Ergebnissen mit einem vom Kunden verwalteten Schlüssel AWS KMS

Sie können optional einen vom Kunden verwalteten AWS KMS Schlüssel angeben, um Abfrageergebnisse zu verschlüsseln, die mit SSE-KMS an Amazon S3 übermittelt werden. Der AWS KMS Schlüssel kann sich in demselben Konto wie die geplante Abfrage oder in einem anderen Konto befinden.

Wenn Sie einen AWS KMS Schlüssel angeben, verwendet die geplante Abfrage diesen Schlüssel, um Ergebnisse über SSE-KMS zu verschlüsseln. Wenn Sie keinen AWS KMS Schlüssel angeben, gelten die Standardverschlüsselungseinstellungen des Buckets. Wenn der Bucket mit der standardmäßigen SSE-KMS-Verschlüsselung unter Verwendung eines vom Kunden verwalteten Schlüssels konfiguriert ist, muss die IAM-Rolle für die Zielzustellung dennoch über `kms:GenerateDataKey` Berechtigungen für diesen Schlüssel verfügen.

Für die IAM-Rolle für die Zielzustellung ist eine `kms:GenerateDataKey` Genehmigung für den Schlüssel erforderlich. AWS KMS Das folgende Beispiel zeigt die erforderlichen Berechtigungen für ein Amazon S3 S3-Ziel mit einem vom Kunden verwalteten AWS KMS Schlüssel:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::my-bucket/prefix/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "kms:GenerateDataKey",
      "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-id",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "s3.us-east-1.amazonaws.com"
        },
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::my-bucket*"
        }
      }
    }
  ]
}
```

Wenn sich der AWS KMS Schlüssel in einem anderen Konto als der IAM-Rolle für die Zielzustellung befindet, muss die AWS KMS Schlüsselrichtlinie im Konto, das den Schlüssel besitzt, der Rolle explizit Zugriff gewähren:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowScheduledQueryRoleToEncrypt",
```

```

    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::111122223333:role/my-s3-delivery-role"
    },
    "Action": "kms:GenerateDataKey",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:ViaService": "s3.us-east-1.amazonaws.com"
      },
      "StringLike": {
        "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::my-bucket*"
      }
    }
  }
]
}

```

Note

Wenn sich der AWS KMS Schlüssel und die IAM-Rolle für die Zielzustellung in demselben Konto befinden, reicht die IAM-Identitätsrichtlinie allein aus, wenn die AWS KMS Schlüsselrichtlinie die Standardanweisung „IAM-Richtlinien aktivieren“ enthält. Eine ausdrückliche Erteilung der AWS KMS Schlüsselrichtlinie ist nur erforderlich, wenn die Schlüsselrichtlinie nicht an IAM delegiert wird.

Die IAM-Rolle für die Veröffentlichung von Abfrageergebnissen in Amazon S3 muss getrennt von der IAM-Rolle für die geplante Abfrageausführung konfiguriert werden. Diese Trennung ermöglicht eine differenzierte Zugriffskontrolle, bei der die Ausführungsrolle Abfragen ausführen kann, während die Amazon S3 S3-Rolle speziell die Ergebnislieferung übernimmt. Beide Rollen müssen eine Vertrauensrichtlinie enthalten, die es dem CloudWatch Logs-Service (`logs.amazonaws.com`) ermöglicht, die Rolle zu übernehmen.

Problembehandlung bei geplanten Abfragen

Verwenden Sie diese Themen zur Problembehandlung, um häufig auftretende Probleme mit geplanten Abfragen zu lösen.

Die Ausführung der Abfrage schlägt aufgrund von Berechtigungsfehlern fehl

Beheben Sie Berechtigungsfehler, die verhindern, dass geplante Abfragen ausgeführt oder Ergebnisse an Ziele übermittelt werden.

Berechtigungsfehler treten auf, wenn der Ausführungsrolle die erforderlichen Berechtigungen zum Lesen aus Protokollgruppen oder zum Schreiben in Zielressourcen fehlen.

Um Berechtigungsfehler zu beheben

1. Stellen Sie sicher, dass die Ausführungsrolle über die `logs:DescribeLogGroups` Berechtigungen `logs:StartQuerylogs:GetQueryResults`, und für die Zielprotokollgruppen verfügt.
2. Stellen Sie sicher, dass die Ausführungsrolle über Schreibberechtigungen für die Zielressourcen (z. B. `s3:PutObject` für S3-Buckets) verfügt.
3. Vergewissern Sie sich, dass die Vertrauensrichtlinie CloudWatch Logs erlaubt, die Ausführungsrolle zu übernehmen. Die Rolle sollte dem Prinzipal (`logs.amazonaws.com`) des Protokolldienstes in ihrer Vertrauensrichtlinie vertrauen.

Zu den häufigsten Ursachen gehören fehlende IAM-Berechtigungen, falsche Ressourcen ARNs in der Richtlinie oder Probleme mit der Konfiguration der Vertrauensrichtlinie.

Um Berechtigungsfehler zu vermeiden, sollten Sie bei der Erstellung von Ausführungsrollen und Testberechtigungen das Prinzip der geringsten Rechte verwenden, bevor Sie geplante Abfragen in der Produktion bereitstellen.

Zeitlimit für Abfragen überschritten

Beheben Sie Timeoutfehler, die auftreten, wenn geplante Abfragen das maximale Ausführungszeitlimit überschreiten.

Abfrage-Timeouts treten auf, wenn die Abfrage mehr als 60 Minuten benötigt, um den angegebenen Datenbereich zu verarbeiten, was häufig auf große Datenmengen oder eine komplexe Abfragelogik zurückzuführen ist.

Um Timeout-Fehler zu beheben

1. Reduzieren Sie den Zeitbereich, indem Sie den Startzeitversatz verringern, um weniger Daten pro Ausführung zu verarbeiten.

2. Optimieren Sie die Abfrage, indem Sie zu Beginn der Abfrage Filter hinzufügen, um die Menge der verarbeiteten Daten zu reduzieren. Verwenden Sie Filterindizes, um die Größe des Datenscans zu reduzieren.
3. Erwägen Sie, komplexe Abfragen in einfachere, zielgerichtete Abfragen aufzuteilen.

Zu den häufigsten Ursachen gehören die Abfrage großer Zeitbereiche, die Verarbeitung von Protokollgruppen mit hohem Volumen oder die Verwendung komplexer Aggregationen ohne angemessene Filterung.

Um Zeitüberschreitungen zu vermeiden, testen Sie Abfragen manuell in CloudWatch Logs Insights mit Ihrem erwarteten Datenvolumen und optimieren Sie die Leistung vor der Planung.

Die Zielverarbeitung schlägt fehl

Beheben Sie Fehler, die auftreten, wenn geplante Abfrageergebnisse nicht an die konfigurierten Ziele übermittelt werden können.

Fehler bei der Zielverarbeitung treten auf, wenn auf den Amazon S3 S3-Ziel-Bucket oder EventBridge Event-Bus nicht zugegriffen werden kann oder der Zielbus falsch konfiguriert ist.

Um Fehler zu beheben, bei denen Abfrageergebnisse nicht im Ziel veröffentlicht werden

1. Stellen Sie sicher, dass der angegebene Amazon S3 S3-Bucket existiert und darauf zugegriffen werden kann.
2. Überprüfen Sie, ob die Zielkonfiguration korrekt ist URIs.
3. Stellen Sie sicher, dass die Ausführungsrolle über die erforderlichen Berechtigungen verfügt, um in das Ziel zu schreiben.

Zu den häufigsten Ursachen gehören gelöschte oder umbenannte Zielressourcen, falsche Ziele URIs oder Probleme mit der Netzwerkkonnektivität.

Überprüfen Sie die Zielkonfigurationen regelmäßig und überwachen Sie die Verfügbarkeit der Zielressourcen, um Zielausfälle zu vermeiden.

Ungültige Abfragefehler

Beheben Sie Syntax- und Logikfehler in geplanten Abfragezeichenfolgen, die eine erfolgreiche Ausführung verhindern.

Ungültige Abfragefehler treten auf, wenn die Abfragezeichenfolge Syntaxfehler enthält, auf nicht vorhandene Felder verweist oder Funktionen der Abfragesprache verwendet, die nicht unterstützt werden.

Um ungültige Abfragefehler zu beheben

1. Testen Sie Ihre Abfrage manuell in CloudWatch Logs Insights, um Syntax und Logik zu überprüfen.
2. Überprüfen Sie, ob alle Protokollfelder, auf die verwiesen wird, in Ihren Zielprotokollgruppen vorhanden sind.
3. Stellen Sie sicher, dass die von Ihnen verwendeten Abfragesprachenfunktionen für geplante Abfragen unterstützt werden.

Zu den häufigsten Ursachen gehören Tippfehler in Feldnamen, falsche Abfragesyntax oder die Verwendung von Abfragefunktionen, die in der Umgebung für die geplante Ausführung nicht unterstützt werden.

Um ungültige Abfragefehler zu vermeiden, sollten Sie Abfragen vor der Planung immer interaktiv testen und Feldnamen mithilfe von Felderkennungsfunktionen überprüfen.

Fehler bei der Parallelität von Abfragen

Im Folgenden sind einige wichtige Punkte zu beachten, wenn Parallelitätsfehler auftreten, da geplante Abfragen dasselbe Kontingent verwenden wie Cloudwatch Logs Insights-Abfragen. Es wird empfohlen, Ihre Zeitpläne zu verteilen, um zu vermeiden, dass das Parallelitätslimit überschritten wird.

- **Kontingent:** Sie können bis zu 100 CloudWatch Logs Insights-Abfragen gleichzeitig pro AWS Konto ausführen.
- **Dashboards:** Zu CloudWatch Dashboards hinzugefügte Abfragen werden ebenfalls auf dieses Parallelitätslimit angerechnet, da sie ausgeführt werden, wenn das Dashboard geladen oder aktualisiert wird.
- **OpenSearch Service PPL/SQL:** Sie können bis zu 15 PPL- oder SQL-Abfragen gleichzeitig OpenSearch pro Konto ausführen. OpenSearch AWS
- **Kontoübergreifende Abfragen:** Das Kontingent für Parallelität gilt sowohl für einzelne als auch für kontenübergreifende Abfragen. Bei Verwendung CloudWatch kontenübergreifender Observability

werden Abfragen, die in einem Überwachungskonto für ein verknüpftes Quellkonto initiiert wurden, ebenfalls auf die Parallelitätsgrenze des Überwachungskontos angerechnet.

- Protokollgruppen mit seltenem Zugriff: Für Protokollgruppen der Klasse der Logs mit seltenem Zugriff ist die maximale Anzahl gleichzeitiger Logs Insights-Abfragen auf fünf begrenzt.

Erkennung von Protokollanomalien

Sie können Anomalien in Ihren Protokolldaten auf zwei Arten erkennen: indem Sie einen Log-Anomalie-Detektor für die kontinuierliche Überwachung erstellen oder indem Sie den [anomaly detection](#) Befehl in CloudWatch Logs Insights-Abfragen für Analysen auf Abruf verwenden.

Ein Detektor für Protokollanomalien scannt die in eine Protokollgruppe aufgenommenen Protokollereignisse und findet automatisch Anomalien in den Protokolldaten. Bei der Erkennung von Anomalien werden maschinelles Lernen und Mustererkennung eingesetzt, um Ausgangswerte für typische Protokollinhalte festzulegen. Für On-Demand-Analysen können Sie den `anomaly detection` Befehl in CloudWatch Logs Insights-Abfragen verwenden, um ungewöhnliche Muster in Zeitreihendaten zu identifizieren. Weitere Informationen zur abfragebasierten Erkennung von Anomalien finden Sie unter: [Verwendung der Anomalieerkennung in Logs Insights CloudWatch](#)

Nachdem Sie einen Anomaliedetektor für eine Protokollgruppe erstellt haben, trainiert er anhand der Protokollereignisse der letzten zwei Wochen in der Protokollgruppe. Die Trainingszeit kann bis zu 15 Minuten dauern. Nach Abschluss der Schulung werden eingehende Protokolle analysiert, um Anomalien zu identifizieren. Die Anomalien werden dann in der CloudWatch Protokollkonsole angezeigt, sodass Sie sie untersuchen können.

CloudWatch Die Protokollmustererkennung extrahiert Protokollmuster, indem statische und dynamische Inhalte in Ihren Protokollen identifiziert werden. Muster sind nützlich für die Analyse großer Protokollsätze, da eine große Anzahl von Protokollereignissen oft in wenige Muster komprimiert werden kann.

Sehen Sie sich zum Beispiel das folgende Beispiel mit drei Protokollereignissen an.

```
2023-01-01 19:00:01 [INFO] Calling DynamoDB to store for ResourceID: 12342342k124-12345
2023-01-01 19:00:02 [INFO] Calling DynamoDB to store for ResourceID: 324892398123-1234R
2023-01-01 19:00:03 [INFO] Calling DynamoDB to store for ResourceID: 3ff231242342-12345
```

Im vorherigen Beispiel folgen alle drei Protokollereignisse einem Muster:

```
<Date-1> <Time-2> [INFO] Calling DynamoDB to store for resource id <ResourceID-3>
```

Felder innerhalb eines Musters werden als Token bezeichnet. Felder, die innerhalb eines Musters variieren, z. B. eine Anforderungs-ID oder ein Zeitstempel, werden als dynamische Token bezeichnet. Jeder unterschiedliche Wert, der für ein dynamisches Token gefunden wird, wird als Tokenwert bezeichnet.

Wenn CloudWatch Logs den Datentyp ableiten kann, für den ein dynamisches Token steht, wird das Token als `<string-number>` angezeigt. Das `string` ist eine Beschreibung des Datentyps, für den das Token steht. Das `number` zeigt, an welcher Stelle des Musters dieses Token im Vergleich zu den anderen dynamischen Tokens erscheint.

CloudWatch Logs weist die Zeichenfolge als Teil des Namens auf der Grundlage der Analyse des Inhalts der Protokollereignisse zu, in denen der Name enthalten ist.

Wenn CloudWatch Logs nicht ableiten kann, für welche Art von Daten ein dynamisches Token steht, zeigt es das Token als `<Token-number> number` an und gibt an, wo im Muster dieses Token im Vergleich zu den anderen dynamischen Tokens vorkommt.

Zu den häufigsten Beispielen für dynamische Token gehören Fehlercodes, IP-Adressen, Zeitstempel und Anfragen. IDs

Die Erkennung von Protokollanomalien verwendet diese Muster, um Anomalien zu finden. Nach der Trainingszeit für das Modell des Anomaliedetektors werden die Protokolle anhand bekannter Trends bewertet. Der Anomaliedetektor kennzeichnet signifikante Schwankungen als Anomalien.

In diesem Kapitel wird beschrieben, wie Sie die Anomalieerkennung aktivieren, Anomalien anzeigen, Alarme für Log-Anomaliedetektoren erstellen und Metriken zur Veröffentlichung von Protokollanomaliedetektoren erstellen. Außerdem wird beschrieben, wie der Anomaliedetektor und seine Ergebnisse mit verschlüsselt werden können. AWS Key Management Service

Für die Erstellung von Protokollanomaliedetektoren fallen keine Gebühren an.

Schweregrad und Priorität von Anomalien und Mustern

Jeder Anomalie, die von einem Log-Anomaliedetektor gefunden wird, wird eine Priorität zugewiesen. Jedem gefundenen Muster wird ein Schweregrad zugewiesen.

- Die Priorität wird automatisch berechnet und basiert sowohl auf dem Schweregrad des Musters als auch auf dem Ausmaß der Abweichung von den erwarteten Werten. Wenn beispielsweise ein bestimmter Token-Wert plötzlich um 500% steigt, kann diese Anomalie als HIGH Priorität eingestuft werden, auch wenn ihr Schweregrad NONE
- Der Schweregrad basiert nur auf Schlüsselwörtern, die in den Mustern wie FATALERROR, und vorkommen. WARN Wenn keines dieser Schlüsselwörter gefunden wird, wird der Schweregrad eines Musters als gekennzeichnet NONE.

Sichtbarkeit und Zeit der Anomalie

Wenn Sie einen Anomaliedetektor erstellen, geben Sie die maximale Sichtbarkeitsdauer für Anomalien an. Dies ist die Anzahl der Tage, an denen die Anomalie in der Konsole angezeigt und durch den [ListAnomalies](#) API-Vorgang zurückgegeben wird. Tritt eine Anomalie nach Ablauf dieses Zeitraums weiterhin auf, wird sie automatisch als normales Verhalten akzeptiert und das Anomaliedetektormodell meldet sie nicht mehr als Anomalie.

Wenn Sie die Sichtbarkeitszeit bei der Erstellung eines Anomaliedetektors nicht anpassen, werden 21 Tage als Standard verwendet.

Unterdrückung einer Anomalie

Nachdem eine Anomalie gefunden wurde, können Sie wählen, ob sie vorübergehend oder dauerhaft unterdrückt werden soll. Das Unterdrücken einer Anomalie bewirkt, dass der Anomaliedetektor dieses Ereignis für den von Ihnen angegebenen Zeitraum nicht mehr als Anomalie kennzeichnet. Wenn Sie eine Anomalie unterdrücken, können Sie wählen, ob Sie nur diese spezifische Anomalie oder alle Anomalien unterdrücken möchten, die mit dem Muster zusammenhängen, in dem die Anomalie gefunden wurde.

Sie können unterdrückte Anomalien weiterhin in der Konsole anzeigen. Sie können sich auch dafür entscheiden, sie nicht mehr zu unterdrücken.

Häufig gestellte Fragen

AWS Verwendet es meine Daten, um Algorithmen für maschinelles Lernen für den AWS Gebrauch oder für andere Kunden zu trainieren?

Nein. Das durch das Training erstellte Modell zur Erkennung von Anomalien basiert auf den Protokollereignissen in einer Protokollgruppe und wird nur innerhalb dieser Protokollgruppe und dieses AWS Kontos verwendet.

Welche Arten von Protokollereignissen eignen sich gut für die Erkennung von Anomalien?

Die Erkennung von Protokollanomalien eignet sich gut für: Anwendungsprotokolle und andere Protokolltypen, bei denen die meisten Protokolleinträge typischen Mustern entsprechen. Protokollgruppen mit Ereignissen, die Schlüsselwörter für die Protokollebene oder den Schweregrad enthalten, wie INFO, ERROR und DEBUG, eignen sich besonders gut für die Erkennung von Protokollanomalien.

Die Erkennung von Protokollanomalien ist nicht geeignet für: Protokollereignisse mit extrem langen JSON-Strukturen, wie z. B. Logs. CloudTrail Die Musteranalyse analysiert nur die ersten 1500 Zeichen einer Protokollzeile, sodass alle Zeichen, die diese Grenze überschreiten, übersprungen werden.

Audit- oder Zugriffsprotokolle, wie z. B. VPC-Flow-Logs, werden bei der Erkennung von Anomalien ebenfalls weniger Erfolg haben. Die Anomalieerkennung dient der Erkennung von Anwendungsproblemen und ist daher möglicherweise nicht für Netzwerk- oder Zugriffsanomalien geeignet.

Um festzustellen, ob ein Anomaliedetektor für eine bestimmte Protokollgruppe geeignet ist, verwenden Sie die CloudWatch Protokollmusteranalyse, um die Anzahl der Muster in den Protokollereignissen in der Gruppe zu ermitteln. Wenn die Anzahl der Muster nicht mehr als etwa 300 beträgt, funktioniert die Erkennung von Anomalien möglicherweise gut. Weitere Informationen zur Musteranalyse finden Sie unter [Musteranalyse](#).

Was wird als Anomalie gekennzeichnet?

Die folgenden Ereignisse können dazu führen, dass ein Protokollereignis als Anomalie gekennzeichnet wird:

- Ein Protokollereignis mit einem Muster, das in der Protokollgruppe noch nie zuvor beobachtet wurde.
- Eine signifikante Abweichung von einem bekannten Muster.
- Ein neuer Wert für ein dynamisches Token mit einem diskreten Satz üblicher Werte.
- Eine starke Änderung der Häufigkeit, mit der ein Wert für ein dynamisches Token vorkommt.

Obwohl alle oben genannten Punkte möglicherweise als Anomalien gekennzeichnet werden, bedeuten sie nicht alle, dass die Anwendung schlecht funktioniert. Beispielsweise könnten eine higher-than-usual Reihe von 200 Erfolgswerten als Anomalie gekennzeichnet werden. In Fällen wie diesem könnten Sie erwägen, diese Anomalien zu unterdrücken, die nicht auf Probleme hinweisen.

Was passiert mit sensiblen Daten, die maskiert werden?

Alle Teile von Protokollereignissen, die als sensible Daten maskiert sind, werden nicht auf Anomalien überprüft. Weitere Informationen zum Maskieren vertraulicher Daten finden Sie unter [Hilfe zum Schutz vertraulicher Protokolldaten](#) durch Maskierung.

Verwendung der Anomalieerkennung in Logs Insights CloudWatch

Neben der Erstellung von Protokollanomaliedetektoren für die kontinuierliche Überwachung können Sie den `anomaly` Befehl auch in CloudWatch Logs Insights-Abfragen verwenden, um bei Bedarf ungewöhnliche Muster in Ihren Protokolldaten zu identifizieren. Dieser Befehl erweitert die bestehende `pattern` Funktionalität und verwendet maschinelles Lernen, um fünf Arten von Anomalien zu erkennen, darunter Änderungen der Musterhäufigkeit, neue Muster und Token-Variationen.

Der `anomaly` Befehl ist besonders nützlich für:

- Ad-hoc-Analyse historischer Protokolldaten zur Identifizierung ungewöhnlicher Muster
- Untersuchung bestimmter Zeiträume auf anomales Verhalten
- Überwachung von Anwendungen wie Lambda-Funktionen auf Ausführungsprobleme

Weitere Hinweise zur Verwendung des `anomaly` Befehls in Ihren Abfragen finden Sie unter [Anomalie](#).

Diese abfragebasierte Anomalieerkennung ergänzt die in den folgenden Abschnitten beschriebenen kontinuierlichen Anomaliedetektoren und bietet Ihnen sowohl Funktionen zur Echtzeitüberwachung als auch zur Analyse auf Abruf.

Aktivieren Sie die Anomalieerkennung für eine Protokollgruppe

Gehen Sie wie folgt vor, um mithilfe der CloudWatch Konsole einen Detektor für Protokollanomalien zu erstellen, der eine Protokollgruppe auf Anomalien durchsucht.

Sie können Anomaliedetektoren auch programmgesteuert erstellen. Weitere Informationen finden Sie unter [CreateLogAnomalyDetector](#).

Um einen Log-Anomaliedetektor zu erstellen

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie Logs, Log Anomalies aus.
3. Wählen Sie „Anomaliedetektor erstellen“.
4. Wählen Sie die Protokollgruppe aus, für die dieser Anomaliedetektor erstellt werden soll.
5. Geben Sie im Feld Name des Anomaliedetektors einen Namen für den Detektor ein.

6. (Optional) Ändern Sie die Häufigkeit der Evaluierung von der Standardeinstellung von 5 Minuten. Stellen Sie diesen Wert entsprechend der Häufigkeit ein, mit der die Protokollgruppe neue Protokolle empfängt. Wenn die Protokollgruppe beispielsweise alle 10 Minuten stapelweise neue Protokollereignisse empfängt, ist es möglicherweise angemessen, die Evaluierungshäufigkeit auf 15 Minuten festzulegen.
7. (Optional) Um den Anomaliedetektor so zu konfigurieren, dass er nur in Protokollereignissen, die bestimmte Wörter oder Zeichenketten enthalten, nach Anomalien sucht, wählen Sie Filtermuster.

Geben Sie dann ein Muster in das Feld Filtermuster für die Anomalieerkennung ein.

Weitere Informationen zur Mustersyntax finden Sie unter. [Filtermustersyntax für Metrikfilter, Abonnementfilter, Filterprotokollereignisse und Live Tail](#)

(Optional) Um Ihr Filtermuster zu testen, geben Sie einige Protokollmeldungen in das Feld Ereignismeldungen protokollieren ein und wählen Sie dann Muster testen aus.

8. (Optional) Um den standardmäßigen Zeitraum für die Sichtbarkeit von Anomalien zu ändern oder diesem Anomaliedetektor einen AWS KMS Schlüssel zuzuordnen, wählen Sie Erweiterte Konfiguration.
 - a. Um den Standardzeitraum für die Sichtbarkeit von Anomalien zu ändern, geben Sie einen neuen Wert in das Feld Maximaler Zeitraum für die Sichtbarkeit von Anomalien (Tage) ein.
 - b. Um diesem Anomaliedetektor einen AWS KMS Schlüssel zuzuordnen, geben Sie den ARN in das Feld KMS-Schlüssel ARN ein. Wenn Sie einen Schlüssel zuweisen, werden die von diesem Detektor gefundenen Anomalie-Informationen im Ruhezustand mit dem Schlüssel verschlüsselt. Benutzer müssen über Berechtigungen für diesen Schlüssel und für den Anomaliedetektor verfügen, um Informationen über die gefundenen Anomalien abzurufen.

Sie müssen außerdem sicherstellen, dass der CloudWatch Logs-Dienstprinzipal berechtigt ist, den Schlüssel zu verwenden. Weitere Informationen finden Sie unter [Verschlüsseln Sie einen Anomaliedetektor und seine Ergebnisse mit AWS KMS](#).

9. Wählen Sie „Anomalieerkennung aktivieren“.

Der Anomaliedetektor wird erstellt und beginnt mit dem Training seines Modells auf der Grundlage der Protokollereignisse, die die Protokollgruppe aufnimmt. Nach etwa 15 Minuten ist die Anomalieerkennung aktiv und beginnt, Anomalien zu finden und aufzudecken.

Zeigt gefundene Anomalien an

Nachdem Sie einen oder mehrere Detektoren für Protokollanomalien erstellt haben, können Sie in der CloudWatch Konsole die gefundenen Anomalien anzeigen.

Sie können Anomalien programmgesteuert anzeigen. Weitere Informationen finden Sie unter [ListAnomalies](#).

Um die Anomalien anzuzeigen, die von all Ihren Protokollanomaliedetektoren gefunden wurden

1. Öffnen Sie die Konsole unter CloudWatch . <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie Logs, Log Anomalies aus.

Die Tabelle Protokollanomalien wird angezeigt. Die Zahl oben neben Protokollanomalien gibt an, wie viele Protokollanomalien in der Tabelle aufgeführt sind. Jede Zeile in der Tabelle enthält die folgenden Informationen:

- In der Spalte Anomalie wird eine kurze Zusammenfassung der Anomalie angezeigt. Diese Zusammenfassungen werden anhand von Protokollen generiert. CloudWatch
 - Die Priorität der Anomalie. Die Priorität wird automatisch auf der Grundlage des Umfangs der Änderungen in den Protokollereignissen, Schlüsselwörtern wie „Exception Auftreten in einem Protokollereignis“ und mehr berechnet.
 - Das Protokollmuster, auf dem die Anomalie basiert. Weitere Hinweise zu Mustern finden Sie unter [Erkennung von Protokollanomalien](#).
 - Der Trend im Anomalieprotokoll zeigt ein Histogramm an, das die Anzahl der Logs darstellt, die dem Muster entsprechen.
 - Letzte Erkennungszeit zeigt den Zeitpunkt an, zu dem diese Anomalie zuletzt gefunden wurde.
 - Die Zeit der ersten Erkennung zeigt an, wann diese Anomalie zum ersten Mal gefunden wurde.
 - Der Anomaliedetektor zeigt den Namen der Protokollgruppe an, die die Protokollereignisse im Zusammenhang mit dieser Anomalie enthält. Sie können diesen Namen wählen, um die Detailseite der Protokollgruppe anzuzeigen.
3. Um eine Anomalie genauer zu untersuchen, wählen Sie das Optionsfeld in der entsprechenden Zeile.

Das Fenster „Pattern Inspect“ wird geöffnet und enthält Folgendes:

- Das Muster, auf dem diese Anomalie basiert. Wählen Sie ein Token innerhalb des Musters aus, um die Werte dieses Tokens zu analysieren.
- Ein Histogramm, das die Häufigkeit des Auftretens der Anomalie im abgefragten Zeitraum zeigt.
- Auf der Registerkarte Protokollbeispiele werden einige der Protokollereignisse angezeigt, die Teil der Anomalie sind.
- Auf der Registerkarte Token-Werte werden die Werte des ausgewählten dynamischen Tokens angezeigt, sofern Sie eines ausgewählt haben.

 Note

Für jedes Token werden maximal 10 Token-Werte erfasst. Die Anzahl der Tokens ist möglicherweise nicht genau. CloudWatch Logs verwendet einen probabilistischen Zähler, um die Tokenanzahl zu generieren, nicht den absoluten Wert.

4. Um eine Anomalie zu unterdrücken, wählen Sie das Optionsfeld in der entsprechenden Zeile und gehen Sie dann wie folgt vor:
 - a. Wählen Sie „Aktionen“, „Anomalie unterdrücken“.
 - b. Geben Sie dann an, wie lange die Anomalie unterdrückt werden soll.
 - c. Um alle Anomalien im Zusammenhang mit diesem Muster zu unterdrücken, wählen Sie „Muster unterdrücken“.
 - d. Wählen Sie „Anomalie unterdrücken“.

Um die in einer einzelnen Protokollgruppe gefundenen Anomalien anzuzeigen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie Protokolle, Protokollgruppen aus.
3. Wählen Sie den Namen einer Protokollgruppe und dann die Registerkarte Anomalieerkennung.

Die Tabelle zur Erkennung von Anomalien wird angezeigt. Die Zahl oben neben Protokollanomalien gibt an, wie viele Protokollanomalien in der Tabelle aufgeführt sind. In jeder Zeile der Tabelle werden die folgenden Informationen angezeigt:

- In der Spalte Anomalie wird eine kurze Zusammenfassung der Anomalie angezeigt. Diese Zusammenfassungen werden anhand von Protokollen generiert. CloudWatch

- Die Priorität der Anomalie. Die Priorität wird automatisch auf der Grundlage des Umfangs der Änderungen in den Protokollereignissen, Schlüsselwörtern wie „Exception Auftreten in einem Protokollereignis“ und mehr berechnet.
 - Das Protokollmuster, auf dem die Anomalie basiert. Weitere Hinweise zu Mustern finden Sie unter [Erkennung von Protokollanomalien](#).
 - Der Trend im Anomalieprotokoll zeigt ein Histogramm an, das die Anzahl der Logs darstellt, die dem Muster entsprechen.
 - Letzte Erkennungszeit zeigt den Zeitpunkt an, zu dem diese Anomalie zuletzt gefunden wurde.
 - Die Zeit der ersten Erkennung zeigt an, wann diese Anomalie zum ersten Mal gefunden wurde.
4. Um eine Anomalie genauer zu untersuchen, wählen Sie das Optionsfeld in der entsprechenden Zeile.

Das Fenster „Pattern Inspect“ wird geöffnet und enthält Folgendes:

- Das Muster, auf dem diese Anomalie basiert. Wählen Sie ein Token innerhalb des Musters aus, um die Werte dieses Tokens zu analysieren.
- Ein Histogramm, das die Häufigkeit des Auftretens der Anomalie im abgefragten Zeitraum zeigt.
- Auf der Registerkarte Protokollbeispiele werden einige der Protokollereignisse angezeigt, die Teil der Anomalie sind.
- Auf der Registerkarte Token-Werte werden die Werte des ausgewählten dynamischen Tokens angezeigt, sofern Sie eines ausgewählt haben.

 Note

Für jedes Token werden maximal 10 Token-Werte erfasst. Die Anzahl der Tokens ist möglicherweise nicht genau. CloudWatch Logs verwendet einen probabilistischen Zähler, um die Tokenanzahl zu generieren, nicht den absoluten Wert.

5. Um eine Anomalie zu unterdrücken, wählen Sie das Optionsfeld in der entsprechenden Zeile und gehen Sie dann wie folgt vor:
- a. Wählen Sie „Aktionen“, „Anomalie unterdrücken“.
 - b. Geben Sie dann an, wie lange die Anomalie unterdrückt werden soll.
 - c. Um alle Anomalien im Zusammenhang mit diesem Muster zu unterdrücken, wählen Sie „Muster unterdrücken“.

- d. Wählen Sie „Anomalie unterdrücken“.

Erstellen Sie Alarme für Protokollanomaliedetektoren

Sie können einen Alarm für einen Melder für Protokollanomalien in einer Protokollgruppe erstellen. Sie können festlegen, dass der Alarm in den ALARM Status wechselt, wenn innerhalb eines bestimmten Zeitraums eine bestimmte Anzahl von Anomalien in der Protokollgruppe gefunden wird. Sie können auch Filter verwenden, sodass nur Anomalien mit einer bestimmten Priorität vom Alarm gezählt werden.

Um einen Alarm für einen Log-Anomaliedetektor zu erstellen

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im Navigationsbereich Logs, Log Anomalies aus.

Die Tabelle der Detektoren für Protokollanomalien wird angezeigt.

3. Wählen Sie das Optionsfeld für den Anomaliedetektor, für den Sie den Alarm einstellen möchten, und wählen Sie Alarm erstellen.

Der Assistent zur CloudWatch Alarmerstellung wird angezeigt. In dem LogAnomalyDetectorFeld wird der Name des ausgewählten Anomaliedetektors angezeigt. Das Feld Metrikname wird angezeigt AnomalyCount.

4. (Optional) Gehen Sie wie folgt vor, um diesen Alarm nach der Priorität von Anomalien zu filtern:
 - Damit der Alarm nur Anomalien mit hoher Priorität zählt, geben Sie für ein. **HIGH** LogAnomalyPriority
 - Damit der Alarm nur Anomalien mit hoher und mittlerer Priorität zählt, geben Sie für ein. **MEDIUM** LogAnomalyPriority

Weitere Informationen zu Prioritätsstufen finden Sie unter. [Schweregrad und Priorität von Anomalien und Mustern](#)

5. Wählen Sie, ob Sie einen statischen oder einen metrischen Schwellenwert für die Erkennung von Anomalien für den Alarm verwenden möchten. Diese Auswahl bestimmt, wie der Alarmschwellenwert festgelegt wird. Ein statischer Schwellenwert bedeutet, dass der Alarmschwellenwert eine statische, konstante Zahl ist, die Sie wählen. Ein Schwellenwert für die Erkennung von Anomalien CloudWatch bestimmt einen Bereich üblicher Werte und der Alarm wird ausgelöst, wenn die tatsächliche Anzahl den Schwellenwert dieses Bandes überschreitet.

Sie müssen nicht die Option „Anomalieerkennung“ für einen Alarm zur Protokollierung von Anomalieerkennung wählen. Weitere Informationen zur Erkennung metrischer Anomalien finden Sie unter [Verwenden CloudWatch](#) der Anomalieerkennung.

6. Für wann auch ***your-metric-name*** immer. , wählen Sie Größer, Größer/Gleich, Niedriger/ Gleich oder Niedriger. Geben Sie danach für als ... eine Zahl für Ihren Schwellenwert an. Der Alarm geht in den **ALARM** Status über, wenn der Anomaliendetektor innerhalb eines im Parameter Zeitraum angegebenen Zeitraums mehr als diese Anzahl von Alarmen findet.
7. Wählen Sie Additional configuration (Zusätzliche Konfiguration). Geben Sie unter Datapoints to alarm (Datenpunkte für Alarm) an, wie viele Auswertungszeiträume (Datenpunkte) im Status ALARM sein müssen, damit der Alarm ausgelöst wird. Wenn die beiden Werte hier übereinstimmen, erstellen Sie einen Alarm, der in den Status ALARM wechselt, wenn entsprechend viele aufeinanderfolgende Zeiträume überschritten werden.

Um einen M-aus-N-Alarm zu erzeugen, geben Sie für den ersten Wert eine Zahl an, die niedriger ist als die Zahl für den zweiten Wert. Weitere Informationen finden Sie unter [Einen Alarm auswerten](#).

8. Wählen Sie für Fehlende Datenverarbeitung aus, wie sich der Alarm verhalten soll, wenn einige Datenpunkte fehlen. Weitere Informationen finden Sie unter [Konfiguration der Behandlung fehlender Daten durch CloudWatch Alarme](#).
9. Wählen Sie Weiter aus.
10. Wählen Sie unter Benachrichtigung die Option Benachrichtigung hinzufügen und geben Sie dann ein Amazon SNS SNS-Thema an, um Sie zu benachrichtigen, wenn Ihr Alarm in den INSUFFICIENT_DATA Status ALARMOK, oder übergeht.
 - a. (Optional) Um mehrere Benachrichtigungen für den gleichen Alarmstatus oder für verschiedene Statuswerte zu senden, wählen Sie Add notification (Benachrichtigung hinzufügen) aus.

 Note

Wir empfehlen, dass Sie den Alarm so einstellen, dass er zusätzlich zu dem Alarm-Zustand auch dann Maßnahmen ergreift, wenn er in den Status Ungenügend Daten wechselt. Dies liegt daran, dass viele Probleme mit der Lambda-Funktion, die eine Verbindung zur Datenquelle herstellt, dazu führen können, dass der Alarm auf Unzureichende Daten übergeht.

- b. (Optional) Damit keine Amazon-SNS-Benachrichtigungen gesendet werden, wählen Sie Entfernen.
11. (Optional) Wenn Sie möchten, dass Ihr Alarm Aktionen für Amazon EC2 Auto Scaling, Amazon EC2, Tickets oder ausführt AWS Systems Manager, wählen Sie die entsprechende Schaltfläche und geben Sie den Alarmstatus und die Aktion an.

 Note

Ihr Alarm kann Aktionen des Systems-Managers nur ausführen, wenn sie in den ALARM-Status wechseln. Informationen zu Systems Manager Manager-Aktionen finden Sie unter [Konfiguration CloudWatch zum Erstellen OpsItems und Erstellen](#) von [Incidents](#).

12. Wählen Sie Weiter aus.
13. Geben Sie unter Name and description (Name und Beschreibung) einen Namen und eine Beschreibung für den Alarm ein und klicken Sie auf Next (Weiter). Der Name darf nur UTF-8-Zeichen und keine ASCII-Kontrolleingabezeichen enthalten. Die Beschreibung kann Markdown-Formatierungen enthalten, die nur auf der Registerkarte Alarmdetails in der CloudWatch Konsole angezeigt werden. Der Markdown kann nützlich sein, um Links zu Runbooks oder anderen internen Ressourcen hinzuzufügen.

 Tip

Der Alarmname darf nur UTF-8-Zeichen enthalten. Er darf keine ASCII-Steuerzeichen enthalten.

14. Bestätigen Sie unter Preview and create (Vorschau und erstellen), dass die Informationen und Bedingungen Ihres Alarms korrekt sind, und wählen Sie dann Create alarm (Alarm erstellen) aus.

Von Protokollanomaliedetektoren veröffentlichte Metriken

CloudWatch Logs veröffentlicht die AnomalyCountMetrik in CloudWatch Metriken. Diese Metrik wird im AWS/Logs Namespace veröffentlicht.

Die AnomalyCountMetrik wird mit den folgenden Dimensionen veröffentlicht:

- LogAnomalyDetector— Der Name des Anomaliedetektors
- LogAnomalyPriority— Die Prioritätsstufe der Anomalie

Verschlüsseln Sie einen Anomaliedetektor und seine Ergebnisse mit AWS KMS

Die Daten des Anomaliedetektors werden in den Protokollen immer verschlüsselt. CloudWatch Standardmäßig verwendet CloudWatch Logs serverseitige Verschlüsselung für die ruhenden Daten. Als Alternative können Sie AWS Key Management Service für diese Verschlüsselung verwenden. Wenn Sie dies tun, erfolgt die Verschlüsselung mit einem AWS KMS Schlüssel. Die Verschlüsselung AWS KMS wird auf der Ebene des Anomaliedetektors aktiviert, indem ein KMS-Schlüssel einem Anomaliedetektor zugeordnet wird.

Important

CloudWatch Logs unterstützt nur symmetrische KMS-Schlüssel. Verwenden Sie keinen asymmetrischen Schlüssel, um die Daten in Ihren Protokollgruppen zu verschlüsseln. Weitere Informationen finden Sie unter [Verwenden von symmetrischen und asymmetrischen Schlüsseln](#).

Einschränkungen

- Um die folgenden Schritte ausführen zu können, benötigen Sie die folgenden Berechtigungen: `kms:CreateKey`, `kms:GetKeyPolicy` und `kms:PutKeyPolicy`.
- Nachdem Sie einen Schlüssel einem Anomaliedetektor zugeordnet oder die Zuordnung aufgehoben haben, kann es bis zu fünf Minuten dauern, bis der Vorgang wirksam wird.
- Wenn Sie CloudWatch Logs den Zugriff auf einen zugehörigen Schlüssel entziehen oder einen zugehörigen KMS-Schlüssel löschen, können Ihre verschlüsselten Daten in CloudWatch Logs nicht mehr abgerufen werden.

Schritt 1: Erstellen Sie einen AWS KMS Schlüssel

Um einen KMS-Schlüssel zu erstellen, verwenden Sie den folgenden Befehl [create-key](#):

```
aws kms create-key
```

Die Ausgabe enthält die Schlüssel-ID und den Amazon-Ressourcennamen (ARN) des Schlüssels. Das Folgende ist eine Beispielausgabe:

```
{
  "KeyMetadata": {
    "Origin": "AWS_KMS",
    "KeyId": "key-default-1",
    "Description": "",
    "KeyManager": "CUSTOMER",
    "Enabled": true,
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "KeyState": "Enabled",
    "CreationDate": 1478910250.94,
    "Arn": "arn:aws:kms:us-west-2:123456789012:key/key-default-1",
    "AWSAccountId": "123456789012",
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ]
  }
}
```

Schritt 2: Festlegen von Berechtigungen auf dem KMS-Schlüssel

Standardmäßig sind alle AWS KMS Schlüssel privat. Nur der Ressourcenbesitzer kann mit ihnen Daten verschlüsseln und entschlüsseln. Der Ressourceninhaber kann jedoch anderen Benutzern und Ressourcen Zugriffsberechtigungen für den KMS-Schlüssel erteilen. Mit diesem Schritt erteilen Sie dem Prinzipal des CloudWatch Logs-Dienstes die Erlaubnis, den Schlüssel zu verwenden. Dieser Dienstprinzipal muss sich in derselben AWS Region befinden, in der der KMS-Schlüssel gespeichert ist.

Als bewährte Methode empfehlen wir, die Verwendung des KMS-Schlüssels auf die von Ihnen angegebenen AWS Konten oder Anomaliedetektoren zu beschränken.

Speichern Sie zunächst die Standardrichtlinie für Ihren KMS-Schlüssel `policy.json` mit dem folgenden [get-key-policy](#) Befehl:

```
aws kms get-key-policy --key-id key-id --policy-name default --output text > ./  
policy.json
```

Öffnen Sie die Datei `policy.json` in einem Texteditor und fügen Sie den in Fettschrift angezeigten Abschnitt aus einer der folgenden Anweisungen hinzu. Sie trennen die vorhandene Anweisung von der neuen Anweisung durch ein Komma. Diese Anweisungen verwenden Condition Abschnitte, um

die Sicherheit des AWS KMS Schlüssels zu erhöhen. Weitere Informationen finden Sie unter [AWS KMS Schlüssel und Verschlüsselungskontext](#).

Der Condition Abschnitt in diesem Beispiel beschränkt die Verwendung des AWS KMS Schlüssels auf das angegebene Konto, er kann jedoch für jeden beliebigen Anomaliedetektor verwendet werden.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Sid": "EnableIAMUserPermissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "kms:*",
      "Resource": ""
    },
    {
      "Sid": "AllowCloudWatchLogsEncryption",
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey*"
      ],
      "Resource": "",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "StringLike": {
          "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:123456789012:anomaly-detector:*"
        },
        "ArnLike": {
```

```

        "aws:SourceArn": "arn:aws:logs:us-
east-1:123456789012:anomaly-detector:*"
    }
},
{
    "Sid": "AllowCloudWatchLogsDescribeKey",
    "Effect": "Allow",
    "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
    },
    "Action": "kms:DescribeKey",
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": "123456789012"
        }
    }
},
{
    "Sid": "AllowCloudWatchLogsReEncryption",
    "Effect": "Allow",
    "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": "123456789012"
        },
        "StringLike": {
            "kms:EncryptionContext:aws-crypto-ec:aws:logs:arn":
"arn:aws:logs:us-east-1:123456789012:anomaly-detector:*"
        },
        "ArnLike": {
            "aws:SourceArn": "arn:aws:logs:us-
east-1:123456789012:anomaly-detector:*"
        }
    }
}

```

```
    },
    {
      "Sid": "AllowCloudWatchLogsDescribeKeyForReEncryption",
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
      },
      "Action": "kms:DescribeKey",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

Fügen Sie abschließend die aktualisierte Richtlinie mit dem folgenden [put-key-policy](#) Befehl hinzu:

```
aws kms put-key-policy --key-id key-id --policy-name default --policy file://
policy.json
```

Schritt 3: Ordnen Sie einen KMS-Schlüssel einem Anomaliedetektor zu

Sie können einen KMS-Schlüssel einem Anomaliedetektor zuordnen, wenn Sie ihn in der Konsole oder mithilfe von oder erstellen. AWS CLI APIs

Schritt 4: Trennen Sie die Zuordnung des Schlüssels zu einem Anomaliedetektor

Nachdem ein Schlüssel einem Anomaliedetektor zugeordnet wurde, können Sie den Schlüssel nicht mehr aktualisieren. Die einzige Möglichkeit, den Schlüssel zu entfernen, besteht darin, den Anomaliedetektor zu löschen und ihn anschließend neu zu erstellen.

Problembehandlung mit CloudWatch Logs Live Tail

CloudWatch Logs Live Tail hilft Ihnen dabei, Vorfälle schnell zu beheben, indem es eine Streaming-Liste mit neuen Protokollereignissen anzeigt, sobald sie aufgenommen werden. Sie können erfasste Protokolle nahezu in Echtzeit anzeigen, filtern und hervorheben, sodass Sie Probleme schnell erkennen und lösen können. Sie können die Protokolle nach von Ihnen angegebenen Begriffen filtern und auch Protokolle hervorheben, die bestimmte Begriffe enthalten, damit Sie schnell finden, wonach Sie suchen.

Bei Live Tail-Sitzungen fallen Kosten pro Minute an. Weitere Informationen zur Preisgestaltung finden Sie auf der Registerkarte Logs unter [Amazon CloudWatch Pricing](#).

Live Tail wird nur für Protokollgruppen der Standard-Protokollklasse unterstützt. Weitere Informationen zu Protokollklassen finden Sie unter [Klassen protokollieren](#).

In den folgenden Abschnitten wird erklärt, wie Sie Live Tail in der Konsole und in der verwenden AWS CLI. Sie können eine Live Tail-Sitzung auch programmatisch starten. Weitere Informationen finden Sie unter [StartLiveTail](#). SDK-Beispiele finden Sie unter [Starten einer Live-Tail-Sitzung mit einem AWS SDK](#).

Sie können Live Tail auch in der verwenden AWS Toolkit for Visual Studio Code. Informationen zum Starten einer Live-Tail-Sitzung über die VS Code Command Palette finden Sie im [Abschnitt Amazon CloudWatch Logs Live Tail](#) im AWS Toolkit for Visual Studio Code Benutzerhandbuch.

Die Live-Tail-Funktion ist in allen kommerziellen AWS [Regionen](#) verfügbar. Es ist in den Regionen China oder AWS GovCloud (USA) nicht verfügbar.

Note

Die `StartLiveTail` API leitet Anfragen mithilfe der SDK-Hostpräfix-Injektion weiter. SDK-Versionen, die vor dem 1. April 2026 veröffentlicht wurden, rotieren `logs.Region.amazonaws.com`, was VPC-Endpunkte nicht unterstützt. SDK-Versionen, die am oder nach dem 1. April 2026 veröffentlicht wurden, rotieren `logs.Region.amazonaws.com`, was VPC-Endpunkte unterstützt. Informationen zum Einrichten eines VPC-Endpunkts für diese API finden Sie unter [VPC-Endpunkt für CloudWatch Logs erstellen](#).

Starten Sie eine Live-Tail-Sitzung mit dem AWS CLI

Der `start-live-tail` AWS CLI Befehl startet eine Live-Tail-Streaming-Sitzung für eine oder mehrere Protokollgruppen in einem Terminal. Eine Live-Tail-Sitzung kann bis zu drei Stunden dauern. Wenn mehr als 500 Protokollereignisse pro Sekunde dem Filter entsprechen, sind die angezeigten Protokollereignisse ein Beispiel für die Gesamtzahl der Protokollereignisse, um ein Tailing-Erlebnis in Echtzeit zu ermöglichen. [Weitere Informationen zu diesem start-live-tail Befehl finden Sie unter start-live-tail](#)

Sie können den in zwei Modi verwenden: `start-live-tail`

- nur drucken — das ist der Standardmodus
- interaktiv

nur drucken

Im `print-only` Modus werden Protokollereignisse auf dem Terminal gestreamt. Jede Sekunde werden im unteren Bereich neue Ereignisse hinzugefügt, wodurch ein fast in Echtzeit ablaufendes Tailing-Erlebnis entsteht, das `tail -f` dem unter Linux ähnelt.

Geben Sie den folgenden Befehl ein, um eine Live-Tail-Sitzung im Nur-Print-Modus zu starten.

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-east-1:111111222222:log-group:my-logs
```

Wenn Sie den Nur-Print-Modus verwenden, können Sie ihn auch über die Pipeline mit anderen Linux-Befehlen weiterleiten, um die Analysemöglichkeiten zu erhöhen. Im folgenden Beispiel werden Protokollereignisse anhand des `error` Schlüsselworts gefiltert und die zweite und vierte Spalte dieser Ereignisse gedruckt, um Ihnen das Extrahieren bestimmter Informationen zu erleichtern.

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-east-1:111111222222:log-group:my-logs --mode print-only | grep "error" | awk '{print $2, $4}'
```

interaktiv

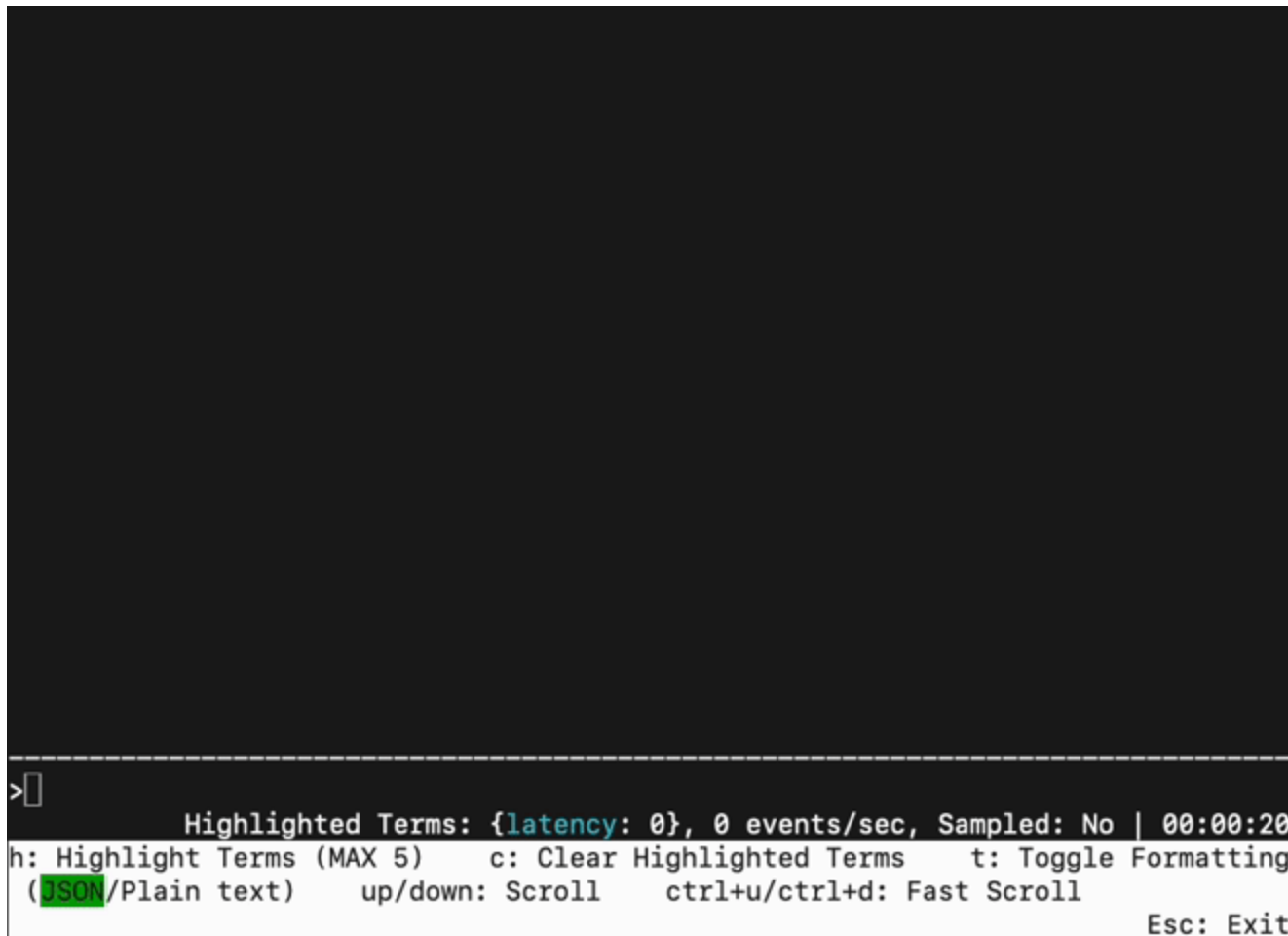
Im `interactive` Modus können Sie Begriffe hervorheben und das Format der ausgegebenen Protokollereignisse zwischen JSON und Klartext umschalten. Im interaktiven Modus werden auch

Informationen über die Live-Tail-Sitzung angezeigt, z. B. die Sitzungsdauer, ob die Sitzung gesampelt wird, und die aktuell hervorgehobenen Begriffe und die Anzahl, mit der sie aufgetreten sind.

Um eine Live-Tail-Sitzung im interaktiven Modus zu starten, geben Sie den folgenden Befehl ein.

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-east-1:111111222222:log-group:my-logs --mode interactive
```

Die Live-Tail-Sitzung beginnt. Das folgende Video zeigt einen Teil einer Beispielsitzung.



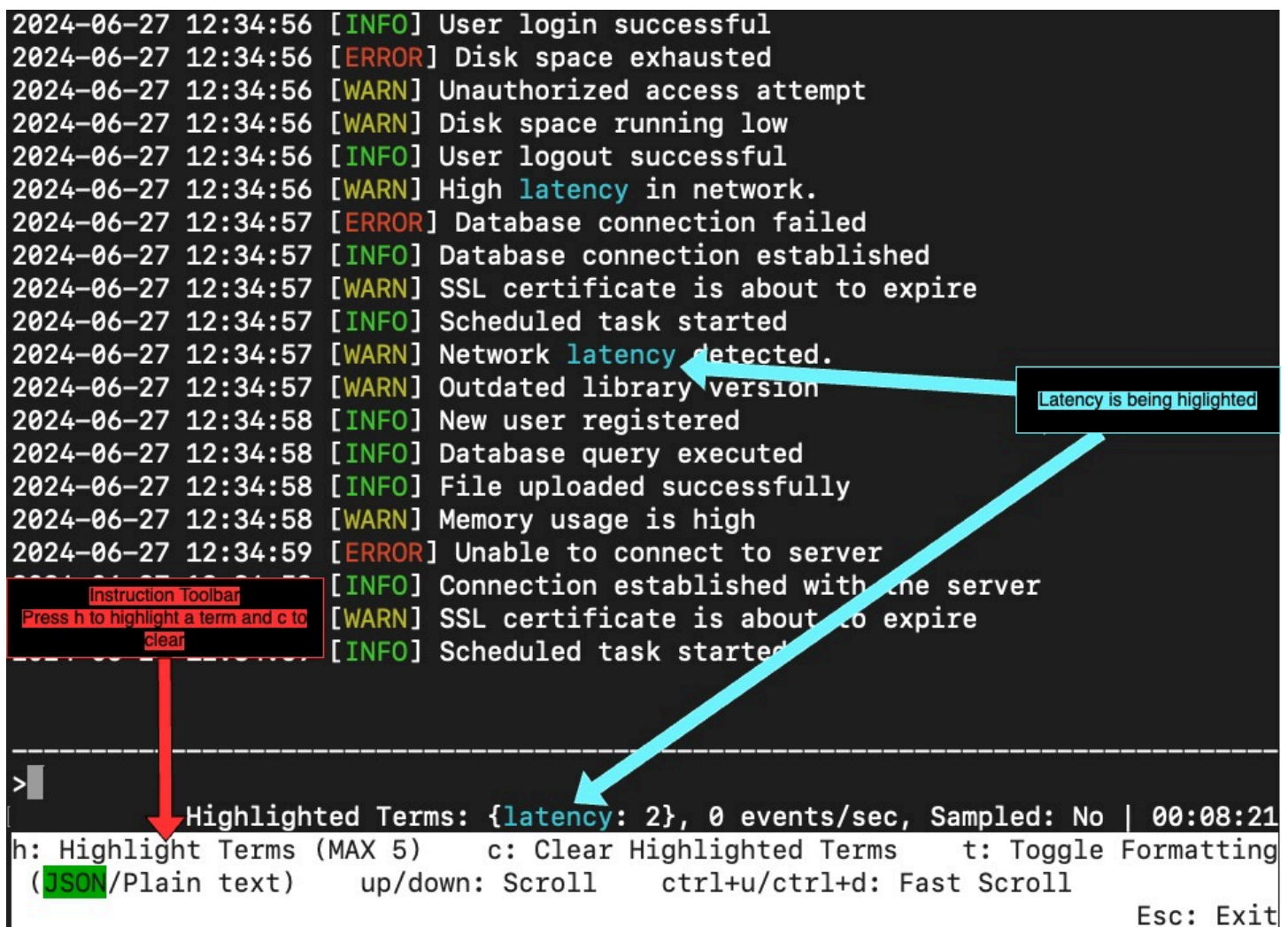
Um einen Begriff in den Streaming-Protokollen hervorzuheben, drücken Sie h und geben Sie dann den Begriff ein. Im Folgenden wird der Bildschirm angezeigt, nachdem der Begriff markiert latency wurde.

Um einen markierten Begriff zu löschen, drücken Sie c und geben dann die Zahl ein, die dem Begriff entspricht, den Sie nicht mehr hervorheben möchten.

Sie können **t** drücken, um das Anzeigeformat eingehender Ereignisse zwischen JSON und Klartext umzuschalten. Diese Umschaltfunktion ist nach bestem Wissen konzipiert und funktioniert nur, wenn das Protokollereignisformat kompatibel ist.

Sie können die Pfeiltasten nach oben und die Pfeiltaste nach unten verwenden, um zu scrollen, CTRL+u und verwenden, CTRL+d um schneller zu scrollen.

Das folgende Bild zeigt die Hervorhebung des `latency` Begriffs während einer Live-Tail-Sitzung.



```
2024-06-27 12:34:56 [INFO] User login successful
2024-06-27 12:34:56 [ERROR] Disk space exhausted
2024-06-27 12:34:56 [WARN] Unauthorized access attempt
2024-06-27 12:34:56 [WARN] Disk space running low
2024-06-27 12:34:56 [INFO] User logout successful
2024-06-27 12:34:56 [WARN] High latency in network.
2024-06-27 12:34:57 [ERROR] Database connection failed
2024-06-27 12:34:57 [INFO] Database connection established
2024-06-27 12:34:57 [WARN] SSL certificate is about to expire
2024-06-27 12:34:57 [INFO] Scheduled task started
2024-06-27 12:34:57 [WARN] Network latency detected.
2024-06-27 12:34:57 [WARN] Outdated library version
2024-06-27 12:34:58 [INFO] New user registered
2024-06-27 12:34:58 [INFO] Database query executed
2024-06-27 12:34:58 [INFO] File uploaded successfully
2024-06-27 12:34:58 [WARN] Memory usage is high
2024-06-27 12:34:59 [ERROR] Unable to connect to server
[INFO] Connection established with the server
[WARN] SSL certificate is about to expire
[INFO] Scheduled task started
```

Instruction Toolbar
Press h to highlight a term and c to clear

Latency is being highlighted

> Highlighted Terms: {latency: 2}, 0 events/sec, Sampled: No | 00:08:21
h: Highlight Terms (MAX 5) c: Clear Highlighted Terms t: Toggle Formatting
(JSON/Plain text) up/down: Scroll ctrl+u/ctrl+d: Fast Scroll Esc: Exit

Starten Sie eine Live-Tail-Sitzung in der Konsole

Sie verwenden die CloudWatch Konsole, um eine Live-Tail-Sitzung zu starten. Das folgende Verfahren erklärt, wie Sie eine Live-Tail-Sitzung über die Option Live Tail im linken Navigationsbereich starten. Sie können Live-Tail-Sitzungen auch von der Seite „Protokollgruppen“ oder der Seite „CloudWatch Logs Insights“ aus starten.

Wenn Sie Datenschutzrichtlinien verwenden, um sensible Daten in einer Protokollgruppe zu maskieren, die Sie mit Live Tail anzeigen, sind die sensiblen Daten in der Live-Tail-Sitzung immer maskiert. Weitere Informationen über die Maskierung sensibler Daten in Protokollgruppen finden Sie unter [Den Schutz vertraulicher Protokolldaten mit Maskierung unterstützen](#).

 Important

Wenn Ihr Netzwerksicherheitsteam die Verwendung von Web-Sockets nicht zulässt, können Sie derzeit nicht auf den Live-Tail-Bereich der CloudWatch Konsole zugreifen. Sie können Live Tail mit den APIs AWS CLI oder verwenden. Weitere Informationen erhalten Sie unter [Starten Sie eine Live-Tail-Sitzung mit dem AWS CLI](#) und [StartLiveTail](#).

Starten einer Live-Tail-Sitzung

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Protokolle, Live Tail aus.
3. Wählen Sie unter Protokollgruppen auswählen die Protokollgruppen aus, deren Ereignisse Sie in der Live-Tail-Sitzung anzeigen möchten. Sie können bis zu 10 Protokollgruppen auswählen.
4. (Optional) Wenn Sie nur eine Protokollgruppe ausgewählt haben, können Sie Ihre Live-Tail-Sitzung weiter filtern, indem Sie einen oder mehrere Protokollstreams auswählen, aus denen Sie Protokollereignisse anzeigen möchten. Wählen Sie dazu unter Protokollstreams auswählen die Namen der Protokollstreams aus der Dropdownliste aus. Alternativ können Sie auch das zweite Feld unter Protokollstreams auswählen verwenden, um ein Präfix für den Namen des Protokollstreams einzugeben. Dann werden alle Protokollstreams ausgewählt, deren Namen mit dem Präfix übereinstimmen.
5. (Optional) Wenn Sie nur Protokollereignisse anzeigen möchten, die bestimmte Wörter oder andere Zeichenfolgen enthalten, geben Sie das Wort oder die Zeichenfolge in Add filter patterns ein.

Um beispielsweise nur Protokollereignisse anzuzeigen, die das Wort **Warning** enthalten, geben Sie **Warning** ein. Im Feld „Filter“ wird zwischen Groß- und Kleinschreibung unterschieden. Sie können mehrere Begriffe und Musteroperatoren in dieses Feld eingeben:

- **error 404** zeigt nur Protokollereignisse an, die sowohl **error** als auch **404** enthalten
- **?Error ?error** zeigt Protokollereignisse an, die entweder **Error** oder **error** enthalten
- **-INFO** zeigt alle Protokollereignisse an, die nicht **INFO** enthalten.

- `{ $.eventType = "UpdateTrail" }` zeigt alle JSON-Protokollereignisse an, bei denen der Wert des Ereignistypfelds `UpdateTrail` lautet

Sie können auch einen regulären Ausdruck (regex) zum Filtern verwenden:

- `%ERROR%` verwendet Regex, um alle Protokollereignisse anzuzeigen, die aus dem Schlüsselwort `ERROR` bestehen
- `{ $.names = %Steve% }` verwendet Regex, um JSON-Protokollereignisse anzuzeigen, bei denen sich `Steve` in der Eigenschaft `"name"` befindet
- `[ w1 = %abc%, w2 ]` verwendet Regex, um durch Leerzeichen getrennte Protokollereignisse anzuzeigen, bei denen das erste Wort `abc` ist

Weitere Informationen zur Mustersyntax finden Sie unter [Filtermustersyntax](#).

6. (Optional) Um einige der angezeigten Protokollereignisse hervorzuheben, geben Sie einen Suchbegriff ein und markieren ihn unter Live Tail. Geben Sie die hervorzuhebenden Begriffe nacheinander ein. Wenn Sie mehrere Begriffe zur Hervorhebung hinzufügen, wird jedem Begriff eine andere Farbe zugewiesen. Ein Markierungsindikator wird links neben jedem Protokollereignis angezeigt, das den angegebenen Begriff enthält, und erscheint auch unter dem Begriff selbst, wenn Sie das Protokollereignis im Hauptfenster erweitern, um das gesamte Protokollereignis anzuzeigen.

Sie können Filter zusammen mit Hervorhebungen verwenden, um Probleme schnell zu beheben. Sie können beispielsweise die Ereignisse so filtern, dass nur die Ereignisse angezeigt werden, die `Error` enthalten, und dann auch die Ereignisse hervorheben, die `404` enthalten.

7. Zum Starten der Sitzung wählen Sie `Filter anwenden`.

Im Fenster werden nun übereinstimmende Protokollereignisse angezeigt. Die folgenden Informationen werden ebenfalls angezeigt:

- Der Timer zeigt an, wie lange die Live-Tail-Sitzung bereits aktiv ist.
- `events/sec` zeigt an, wie viele aufgenommene Protokollereignisse pro Sekunde den von Ihnen festgelegten Filtern entsprechen.
- Um zu verhindern, dass die Sitzung zu schnell scrollt, weil viele Ereignisse den Filtern entsprechen, zeigt CloudWatch Logs möglicherweise nur einige übereinstimmende Ereignisse an. Wenn dies der Fall ist, wird der Prozentsatz der übereinstimmenden Ereignisse, die auf dem Bildschirm angezeigt werden, in `%` angezeigt.

8. Um den Ablauf der Ereignisse zu unterbrechen, um zu untersuchen, was gerade angezeigt wird, wählen Sie eine beliebige Stelle im Ereignisfenster.
9. Während der Sitzung können Sie wie folgt vorgehen, um weitere Details zu den einzelnen Protokollereignissen anzuzeigen.
 - Um den gesamten Text für ein Protokollereignis im Hauptfenster anzuzeigen, wählen Sie den Pfeil neben dem betreffenden Ereignis.
 - Um den gesamten Text für ein Protokollereignis in einem Seitenfenster anzuzeigen, wählen Sie die Lupe + neben dem betreffenden Protokollereignis. Der Ereignisfluss wird angehalten und das Seitenfenster wird angezeigt.

Die Anzeige des Textes eines Protokollereignisses im Seitenfenster kann nützlich sein, um seinen Text mit anderen Ereignissen im Hauptfenster zu vergleichen.

10. Zum Beenden der Live-Tail-Sitzung wählen Sie Stoppen.
11. Um die Sitzung neu zu starten, verwenden Sie optional den Filterbereich, um die Filterkriterien zu ändern, und wählen Sie Filter anwenden. Wählen Sie dann Anfang aus.

Cross-account Regionsübergreifende Protokollzentralisierung

Bei der Datenzentralisierung von Amazon CloudWatch Logs werden Protokolldaten von mehreren Mitgliedskonten in einem Datenspeicher gesammelt. Dabei werden AWS Organizations konto- und regionsübergreifende Zentralisierungsregeln verwendet. Sie definieren die Regeln, mit denen Protokolldaten automatisch aus mehreren Konten und AWS-Regionen in ein zentrales Konto innerhalb Ihrer Organisation repliziert werden. Diese Funktion optimiert die Protokollkonsolidierung und sorgt so für eine verbesserte zentrale Überwachung, Analyse und Einhaltung von Vorschriften in Ihrer gesamten Infrastruktur. AWS

CloudWatch Die Zentralisierung von Protokolldaten bietet Flexibilität bei der Konfiguration, um Betriebs- und Sicherheitsanforderungen zu erfüllen, z. B. die Möglichkeit, während der Regeleinrichtung innerhalb des Zielkontos eine Backup-Region zu konfigurieren, um die Ausfallsicherheit zu erhöhen. Darüber hinaus haben Sie die volle Kontrolle über das Verschlüsselungsverhalten von Protokollgruppen, die aus Quellkonten kopiert wurden, um Daten zu verarbeiten, die ursprünglich mit vom Kunden verwalteten KMS-Schlüsseln verschlüsselt wurden.

Note

Die Funktion zur Zentralisierung von CloudWatch Protokollen verarbeitet nur neue Protokolldaten, die in den Quellkonten eingehen, nachdem Sie die Zentralisierungsregel erstellt haben. Historische Protokolldaten (Protokolle, die vor der Regelerstellung existierten) sind nicht zentralisiert.

Konzepte der Datenzentralisierung

Machen Sie sich mit den folgenden Konzepten vertraut, bevor Sie mit der Zentralisierung von CloudWatch Logs-Daten beginnen:

Regel zur Zentralisierung

Eine Konfiguration, die definiert, wie Protokolldaten von Quellkonten und Regionen auf ein Zielkonto und eine Zielregion repliziert werden. Regeln spezifizieren Quellkriterien und Zieleinstellungen.

Quellkonto

Das AWS Konto, aus dem die Protokolldaten stammen. Protokollereignisse von Quellkonten werden auf der Grundlage der von Ihnen definierten Zentralisierungsregeln auf das Zielkonto repliziert.

Zielkonto

Das AWS Zielkonto, in dem replizierte Protokolldaten gespeichert werden. Dieses Konto dient als zentraler Ort für die Protokollanalyse und -überwachung.

Backup-Region

Eine optionale sekundäre Region innerhalb des Zielkontos, in der Protokolldaten repliziert werden können, um die Ausfallsicherheit zu erhöhen und die Notfallwiederherstellung zu gewährleisten.

Verschlüsselung in Protokollen CloudWatch

Protokollgruppendaten werden in CloudWatch Logs immer verschlüsselt. Standardmäßig verwendet CloudWatch Logs serverseitige Verschlüsselung mit 256-Bit Advanced Encryption Standard Galois/Counter Mode (AES-GCM), um Protokolldaten im Ruhezustand zu verschlüsseln. Als Alternative können Sie den AWS Key Management Service für diese Verschlüsselung verwenden. Weitere Informationen finden Sie in der [Dokumentation CloudWatch Logs Encryption](#).

- Funktionsweise der Verschlüsselung bei der Zentralisierung: Bei der Zentralisierung von CloudWatch Protokollen werden Protokolldaten bei der Aufnahme aktiv von den Quellkonten auf die Zielkonten kopiert. Während dieses Vorgangs bleiben Ihre Daten während der Übertragung mit einem AWS eigenen Serviceschlüssel verschlüsselt. Daten, die sich sowohl in der Quell- als auch in der Zielprotokollgruppe befinden, werden mit der von Ihnen gewählten Verschlüsselungsmethode (vom Kunden verwaltete oder AWS eigene KMS-Schlüssel) verschlüsselt. Wenn Sie den vom Kunden verwalteten KMS-Schlüssel in Ihren Zielprotokollgruppen verwenden, fügen Sie das Tag dem KMS-Schlüssel hinzu, `LogsManaged = true` damit der Centralization Service darauf zugreifen kann.
- Wenn KMS-Berechtigungen erforderlich sind:
 - Wenn Sie in Ihren Quellkonten vom Kunden verwaltete KMS-Schlüssel verwenden, benötigt CloudWatch Logs in den folgenden Beispielszenarien [KMS-Berechtigungen](#):
 - Durchsatzmanagement: Wenn die Durchsatzgrenzen für die Zentralisierung erreicht sind, werden Protokolldaten vorübergehend verschlüsselt mit Ihrem vom Kunden verwalteten KMS-Schlüssel gespeichert, bis Bandbreite verfügbar ist.

- **Datenschutz und Schwärzung:** Wenn für Quellprotokollgruppen Datenschutzrichtlinien aktiviert sind, benötigt CloudWatch Logs Entschlüsselungsberechtigungen, um auf Protokollrohdaten zugreifen zu können, um sie zu zentralisieren.

Important

Zentralisierungsregeln werden vom Verwaltungskonto der AWS Organizations oder vom delegierten Administrator verwaltet. Um vom Kunden verwaltete KMS-encrypted Protokollgruppen von der Zentralisierung auszuschließen, konfigurieren Sie die Regeleinstellungen auf „Mit AWS KMS-Schlüssel verschlüsselte Protokollgruppen nicht zentralisieren“.

Einrichtung der Protokollzentralisierung

Um die CloudWatch Protokollzentralisierung einzurichten, müssen Sie Zentralisierungsregeln konfigurieren, die definieren, wie Protokolldaten von Protokollgruppen in Quellkonten zu Protokollgruppen in Ihrem Zielkonto fließen.

Sobald die Zentralisierungsregel aktiviert ist und Protokollereignisse auf das Zielkonto repliziert wurden, können Sie Metrik-, Abonnement- und Kontofilter für zentralisierte Protokollgruppen mit erweiterten Filterfunktionen erstellen. Diese Filter können auf Protokollereignisse von bestimmten Quellkonten und Regionen abzielen und Informationen zu Quellkonten und Regionen als metrische Dimensionen ausgeben. Weitere Informationen finden Sie unter [Erstellen von Metriken aus Protokollereignissen mithilfe von Filtern](#).

Voraussetzungen

- AWS Organizations muss eingerichtet sein und das Quell- und das Zielkonto müssen beide zur Organisation gehören.
- Der vertrauenswürdige Zugriff muss sowohl für das Verwaltungskonto als auch für das Zielkonto aktiviert sein CloudWatch, damit der Zugriff auf die Protokolldaten möglich ist.

Note

Es wird empfohlen, den vertrauenswürdigen Zugriff über die Konsole zu aktivieren, wodurch die erforderliche serviceverknüpfte Rolle (SLR) automatisch erstellt wird.

Wenn der vertrauenswürdige Zugriff über andere Methoden aktiviert wird, muss die serviceverknüpfte Rolle separat erstellt werden.

Anpassen der Namen der Zielprotokollgruppen

Wenn Sie eine Zentralisierungsregel erstellen, können Sie mithilfe von Attributen anpassen, wie die Namen der Zielprotokollgruppen strukturiert werden. Diese Attribute werden bei der Erstellung von Protokollgruppen automatisch durch tatsächliche Werte ersetzt, sodass Sie Protokolle in Ihrem Zielkonto hierarchisch organisieren können. Standardmäßig wird nur das `${source.logGroup}` Attribut verwendet, das alle Protokollgruppen mit demselben Namen im Zielkonto zusammenführt. Wenn eine Variable nicht aufgelöst werden kann, erbt sie den Wert von ihrer übergeordneten Variablen in der Hierarchie.

Verfügbare Attribute

Sie können die folgenden Attribute in Ihrem Namensmuster für die Zielprotokollgruppe verwenden:

Attribute für den Namen der Zielprotokollgruppe

Attribut	Description
<code>\${source.accountId}</code>	Die AWS Konto-ID, von der das Protokoll stammt.
<code>\${source.region}</code>	Der AWS-Region Ursprung des Protokolls.
<code>\${source.logGroup}</code>	Der ursprüngliche Name der Protokollgruppe aus dem Quellkonto.
<code>\${source.org.id}</code>	Ihre AWS Organizations ID des Quellkontos.
<code>\${source.org.ouId}</code>	Die Organisationseinheiten-ID des Quellkontos
<code>\${source.org.rootId}</code>	Die Root-ID der Organisation
<code>\${source.org.path}</code>	Der vollständige Organisationspfad vom Konto zum Stammverzeichnis

Beispiele

Behalten Sie die ursprüngliche Struktur der Protokollgruppe bei

Pattern: `/centralized/${source.accountId}${source.logGroup}`

Ergebnis: `/centralized/123456789012/aws/lambda/my-function`

Organisieren Sie nach Konto und Region

Pattern: `/centralized/${source.accountId}/${source.region}`

Ergebnis: `/centralized/123456789012/us-east-1`

Organisieren Sie nach Organisationsstruktur

Pattern: `/logs/${source.org.id}/${source.org.ouId}/${source.accountId}`

Ergebnis: `/logs/o-abc123/ou-xyz-12345678/123456789012`

Einfache flache Struktur

Pattern: `/centralized-logs`

Ergebnis: `/centralized-logs`

Best Practices

- Geben Sie die Quellkonto-ID an, um leicht zu erkennen, von welchen Kontoprotokollen sie stammen.
- Geben Sie die Quellregion an, wenn Sie von mehreren Regionen aus zentralisieren.
- Strukturieren Sie die Namen der Zielprotokollgruppen so, dass sie weniger als 512 Zeichen lang sind. CloudWatch Logs erzwingt eine maximale Länge von Protokollgruppennamen von 512 Zeichen.

Eine Zentralisierungsregel erstellen

Gehen Sie wie folgt vor, um eine Zentralisierungsregel zu erstellen, die Protokolldaten von Quellkonten auf Ihr Zielkonto repliziert.

Um eine Zentralisierungsregel zu erstellen

1. Navigieren Sie im Management- oder Delegated Administrator-Konto der Organisation zur CloudWatch Konsole.
2. Wählen Sie Einstellungen aus.
3. Navigieren Sie zur Registerkarte Organisation.
4. Wählen Sie Regel konfigurieren.
5. Geben Sie die Quelldetails an, indem Sie die folgenden Felder festlegen, und wählen Sie dann Weiter aus:
 - a. Name der Zentralisierungsregel: Geben Sie einen eindeutigen Namen für die Zentralisierungsregel ein.
 - b. Quellkonten: Definieren Sie Auswahlkriterien für Quellen, um Konten auszuwählen, aus denen Telemetriedaten zentralisiert werden sollen. Zu den Auswahlkriterien können gehören:
 - Eine Liste der Mitgliedskonten in der Organisation
 - Eine Liste der Organisationseinheiten in der Organisation
 - Die gesamte Organisation

Sie können die Auswahlkriterien in zwei Modi angeben:

- Builder: Eine klickbasierte Oberfläche zur Generierung der Auswahlkriterien für die Quelle
- Editor: Ein frei formatiertes Textfeld zur Angabe der Auswahlkriterien für die Quelle

Unterstützte Syntax für Kriterien zur Quellenauswahl:

- Unterstützte Schlüssel: OrganizationId | OrganizationUnitId | AccountId | *
 - Unterstützte Operatoren: = | IN | OR
- c. Quellregionen: Wählen Sie eine Liste von Regionen aus, um nach den zu zentralisierenden Telemetriedaten zu suchen.
 6. Geben Sie die Zieldetails an, indem Sie die folgenden Felder festlegen, und wählen Sie dann Weiter aus:
 - a. Zielkonto: Wählen Sie ein Konto in der Organisation aus, das als zentrales Ziel für Telemetriedaten dient.

- b. Zielregion: Wählen Sie eine primäre Region aus, in der eine Kopie der zentralen Telemetriedaten gespeichert wird.
 - c. Backup-Region: Wählen Sie optional eine Region aus, in der eine zweite Kopie der zentralen Telemetriedaten gespeichert wird.
7. Geben Sie Telemetriedaten an, indem Sie die folgenden Felder festlegen, und wählen Sie dann Weiter aus:
- a. Protokollgruppen: Wählen Sie eine der folgenden Optionen:
 - Alle Protokollgruppen: Zentralisieren Sie die Protokolle aller Protokollgruppen in den Quellkonten.
 - Protokollgruppe filtern: Zentralisiert die Protokolle aus einer Teilmenge von Protokollgruppen in den Quellkonten, sodass sie den Auswahlkriterien entsprechen. Sie können die Auswahlkriterien in zwei Modi angeben:
 - Builder: Ein auf Auswahlmöglichkeiten basierendes Erlebnis zur Generierung der Auswahlkriterien
 - Editor: Ein frei formatiertes Textfeld zur Angabe der Auswahlkriterien

Es gibt zwei Auswahlkriterien, die Sie zum Filtern von Protokollen verwenden können:

- Auswahlkriterien für Protokollgruppen: Die Auswahlkriterien, die angeben, welche Quellprotokollgruppen zentralisiert werden sollen.
 - Unterstützte Schlüssel: LogGroupName | *
 - Unterstützte Operatoren: = | != | IN | NICHT IN | UND | ODER | WIE | NICHT
- Auswahlkriterien für Datenquellen: Die Auswahlkriterien, die angeben, welche Datenquellen zentralisiert werden sollen.
 - Unterstützte Schlüssel: | DataSourceName DataSourceType
 - Unterstützte Operatoren: = | != | IN | NICHT IN | UND | ODER | WIE | NICHT

Wenn sowohl Auswahlkriterien für Protokollgruppen als auch für Datenquellen angegeben sind, muss ein Protokollereignis beiden Kriterien entsprechen, damit es zentralisiert werden kann.

- b. Verschlüsselte KMS-Protokollgruppe

⚠ Important

CloudWatch Zentralisierungsregeln können keine Protokolle vom Quellkonto an die Zielprotokollgruppen übermitteln, wenn der in der Zentralisierungsregel angegebene KMS-Schlüssel CloudWatch Logs nicht erlaubt, ihn zu verwenden. Wenn Sie in Ihren Zielprotokollgruppen einen vom Kunden verwalteten KMS-Schlüssel verwenden, fügen Sie dem KMS-Schlüssel das Tag `LogsManaged = true` hinzu. Weitere Informationen finden Sie unter [Schritt 2: Festlegen von Berechtigungen auf dem KMS-Schlüssel](#).

Wählen Sie eine der folgenden Optionen:

- Zentralisieren Sie Quellprotokollgruppen, die mit kundenverwalteten KMS-Schlüsseln verschlüsselt sind, mithilfe eines zielspezifischen, kundenverwalteten KMS-Schlüssels: Zentralisieren Sie Protokollereignisse aus Quellprotokollgruppen, die mit kundenverwalteten KMS-Schlüsseln verschlüsselt wurden, in Zielprotokollgruppen, die mit einem vom Kunden verwalteten KMS-Schlüssel im Zielkonto verschlüsselt sind.

Wenn diese Einstellung ausgewählt ist, müssen Sie auch Folgendes festlegen:

- Zielverschlüsselungsschlüssel ARN: ARN des vom Kunden verwalteten KMS-Schlüssels im Zielkonto und in der primären Zielregion, der neu erstellten Zielprotokollgruppen zugeordnet werden soll.
- ARN für den Verschlüsselungsschlüssel für das Backup-Ziel (falls die Backup-Region ausgewählt ist): ARN des vom Kunden verwalteten KMS-Schlüssels im Zielkonto und in der Backup-Zielregion, der mit neu erstellten Zielprotokollgruppen verknüpft werden soll.
- Die Zentralisierung auf unverschlüsselte Zielprotokollgruppen überspringen (optional): Wenn eine Protokollgruppe bereits ohne einen vom Kunden verwalteten KMS-Schlüssel existiert, CloudWatch kann ihre Verschlüsselung nicht aktualisiert werden. Wählen Sie diese Option, um die Zentralisierung von Protokollereignissen aus Quellprotokollgruppen, die mit kundenverwalteten KMS-Schlüsseln verschlüsselt wurden, in Zielprotokollgruppen zu überspringen, die keinem vom Kunden verwalteten KMS-Schlüssel zugeordnet sind.
- Zentralisieren Sie Protokollgruppen, die mit vom Kunden verwalteten KMS-Schlüsseln verschlüsselt wurden, im Zielkonto mit AWS eigenem KMS-Schlüssel: Zentralisieren Sie Protokollereignisse aus Quellprotokollgruppen, die mit vom Kunden verwalteten KMS-

Schlüsseln verschlüsselt wurden, in neu erstellten Zielprotokollgruppen, die mit einem AWS eigenen KMS-Schlüssel verschlüsselt wurden.

- Zentralisieren Sie keine Protokollgruppen, die mit vom Kunden verwalteten KMS-Schlüsseln verschlüsselt sind: Überspringen Sie die Zentralisierung von Protokollereignissen aus Quellprotokollgruppen, die mit vom Kunden verwalteten KMS-Schlüsseln verschlüsselt wurden.

8. Überprüfen Sie die Zentralisierungsregel, nehmen Sie optional Änderungen in letzter Minute vor und wählen Sie Zentralisierungsrichtlinie erstellen aus.

Eine Zentralisierungsregel ändern

Gehen Sie wie folgt vor, um eine bestehende Zentralisierungsregel zu ändern.

Um eine Zentralisierungsregel zu ändern

1. Navigieren Sie im Management- oder Delegated Administrator-Konto der Organisation zur CloudWatch Konsole.
2. Wählen Sie Einstellungen aus.
3. Navigieren Sie zur Registerkarte Organisation.
4. Wählen Sie Regeln verwalten.
5. Wählen Sie die zu aktualisierende Regel aus und klicken Sie auf Bearbeiten.
6. Aktualisieren Sie die Regelkonfiguration nach Bedarf und wählen Sie Weiter, um mit den einzelnen Schritten fortzufahren.
7. Wählen Sie in Schritt 4, Überprüfen und konfigurieren, die Option Zentralisierungsrichtlinie aktualisieren aus.

Eine Zentralisierungsregel anzeigen

Gehen Sie wie folgt vor, um Details einer vorhandenen Zentralisierungsregel anzuzeigen.

Um eine Zentralisierungsregel anzuzeigen

1. Navigieren Sie im Management- oder Delegated Administrator-Konto der Organisation zur CloudWatch Konsole.
2. Wählen Sie Einstellungen aus.

3. Navigieren Sie zur Registerkarte Organisation.
4. Wählen Sie Regeln verwalten.
5. Zeigen Sie eine Liste aller vorhandenen Zentralisierungsregeln an und wählen Sie einen bestimmten Regelnamen aus, um die Details anzuzeigen.

Löschen einer Zentralisierungsregel

Gehen Sie wie folgt vor, um eine bestehende Zentralisierungsregel zu löschen.

Um eine Zentralisierungsregel zu löschen

1. Navigieren Sie im Management- oder Delegated Administrator-Konto der Organisation zur CloudWatch Konsole.
2. Wählen Sie Einstellungen aus.
3. Navigieren Sie zur Registerkarte Organisation.
4. Wählen Sie Regeln verwalten.
5. Wählen Sie die Regel aus, die Sie löschen möchten, und wählen Sie Löschen.
6. Bestätigen Sie den Löschvorgang und wählen Sie Delete (Löschen) aus.

Zentralisierungsregeln für Überwachung und Fehlerbehebung

Sie können den Status und die Leistung Ihrer Zentralisierungsregeln mithilfe von CloudWatch Metriken, der CloudWatch Protokollkonsole und AWS CloudTrail Protokollen überwachen. Auf diese Weise können Sie sicherstellen, dass Protokolldaten erfolgreich repliziert werden, und Probleme mit Ihrer Zentralisierungskonfiguration identifizieren.

CloudWatch Logs bietet:

1. Regelintegrität pro Zentralisierungsregel
 - a. Wählen Sie Einstellungen aus.
 - b. Navigieren Sie zur Registerkarte Organisation.
 - c. Wählen Sie Regeln verwalten.
2. Protokolliert API-Aufrufe mit AWS CloudTrail

3. CloudWatch veröffentlicht außerdem Metriken für die Zentralisierung, einschließlich der Protokollierung replizierter Ereignisse, Fehler und Drosselung. Weitere Informationen zu diesen Metriken und ihren Dimensionen finden Sie unter: [Metriken und Dimensionen der Zentralisierung](#)

Integritätsstatus der Regel Zentralisierung

Jede Zentralisierungsregel hat einen Integritätsstatus, der angibt, ob sie ordnungsgemäß funktioniert. Sie können den Regelstatus über die Konsole oder programmgesteuert mithilfe der API überprüfen.

Zu den Statusangaben für Regeln gehören:

- **HEALTHY:** Die Regel funktioniert normal und repliziert Protokolldaten wie konfiguriert
- **UNHEALTHY:** Bei der Regel sind Probleme aufgetreten und die Daten werden möglicherweise nicht korrekt repliziert
- **PROVISIONING:** Die Zentralisierung für die Organisation wird derzeit eingerichtet.

Wenn eine Regel als UNGESUND gekennzeichnet ist, enthält das `FailureReason` Feld Einzelheiten zu dem spezifischen Problem, das behoben werden muss.

Überwachung von API-Aufrufen für die Zentralisierung mit AWS CloudTrail

AWS CloudTrail protokolliert API-Aufrufe an den Zentralisierungsdienst, sodass Sie Konfigurationsänderungen nachverfolgen und Probleme für Konten beheben können, die Mitglieder Ihres Unternehmens sind. AWS Organizations

Zu den wichtigsten CloudTrail Ereignissen im Zusammenhang mit der Zentralisierung gehören:

- `CreateCentralizationRuleForOrganization`: Wenn eine neue Zentralisierungsregel erstellt wird
- `UpdateCentralizationRuleForOrganization`: Wenn eine bestehende Regel geändert wird
- `DeleteCentralizationRuleForOrganization`: Wenn eine Regel gelöscht wird
- `GetCentralizationRuleForOrganization`: Wenn Regeldetails abgerufen werden
- `ListCentralizationRulesForOrganization`: Wenn Regeln aufgelistet werden

Sie können CloudTrail Protokolle verwenden, um Änderungen an der Zentralisierungskonfiguration zu überprüfen und sie mit Leistungsproblemen oder Replikationsfehlern zu korrelieren.

Empfehlungen zur Überwachung

Um sicherzustellen, dass die Zentralisierung ordnungsgemäß funktioniert, empfehlen wir, CloudWatch Alarmer für wichtige Zentralisierungskennzahlen einzurichten, die wir an Metrics weiterleiten. CloudWatch Diese proaktive Überwachung hilft Ihnen, Probleme frühzeitig zu erkennen und eine zuverlässige Protokollzentralisierung in Ihrem gesamten Unternehmen aufrechtzuerhalten.

Zu den wichtigsten Kennzahlen, die es zu überwachen gilt, gehören:

- **IncomingCopiedBytes:** Überwachen Sie die Menge der Protokolldaten, die erfolgreich auf Ihr Zielkonto repliziert wurden. Ein plötzlicher Rückgang oder das Fehlen dieser Metrik kann auf Zentralisierungsprobleme hinweisen.
- **CentralizationError:** Richten Sie Alarmer für alle Fehler im Zentralisierungsprozess ein, um Probleme schnell zu erkennen und zu lösen.
- **CentralizationThrottled:** Achten Sie auf Drosselungsereignisse, die sich auf die Leistung der Protokollreplikation auswirken könnten.

Eine vollständige Liste der verfügbaren Zentralisierungsmetriken und ihrer Dimensionen finden Sie unter [Metriken und Dimensionen der Zentralisierung](#)

Wenn Protokolle nicht wie erwartet zentralisiert werden, sollten Sie sich die folgenden häufigen Szenarien ansehen, die eine Zentralisierung von Protokollen verhindern können.

Historische Protokolldaten

Die Funktion zur Zentralisierung von CloudWatch Protokollen verarbeitet nur neue Protokolldaten, die in den Quellkonten eingehen, nachdem Sie die Zentralisierungsregel erstellt haben. Historische Protokolldaten (Protokolle, die vor der Regelerstellung existierten) sind nicht zentralisiert.

KMS-Schlüsselberechtigungen

Zentralisierungsregeln können keine Protokolle vom Quellkonto an die Zielprotokollgruppen übermitteln, wenn der in der Zentralisierungsregel angegebene KMS-Schlüssel CloudWatch Logs nicht erlaubt, ihn zu verwenden. Stellen Sie sicher, dass die KMS-Schlüsselrichtlinie Logs die erforderlichen Berechtigungen gewährt. CloudWatch Weitere Informationen finden Sie unter [Schritt 2: Festlegen von Berechtigungen auf dem KMS-Schlüssel](#).

Vom Kunden verwaltete KMS-Schlüsselkonfiguration

Wenn Sie bei der Regelerstellung die Option Protokollgruppen, die mit dem vom Kunden verwalteten KMS-Schlüssel verschlüsselt wurden, nicht zentralisieren ausgewählt haben, werden Protokollereignisse aus Quellprotokollgruppen, die mit dem vom Kunden verwalteten KMS-Schlüssel verschlüsselt wurden, übersprungen und nicht zentralisiert.

Die Zielverschlüsselung stimmt nicht überein

Wenn die Zielprotokollgruppe bereits mit einer anderen KMS-Verschlüsselungskonfiguration als der in der Zentralisierungsregel angegebenen vorhanden ist und die Konfliktlösung auf SKIP gesetzt ist, werden Datensätze gelöscht und es wird ein `DestinationEncryptionMismatch` Fehler ausgegeben. Dies ist beispielsweise der Fall, wenn das Ziel über eine Standardverschlüsselung verfügt, die Regel jedoch einen vom Kunden verwalteten KMS-Schlüssel spezifiziert.

Vertrauenswürdiger Zugriff nicht aktiviert

Vertrauenswürdiger Zugriff muss aktiviert sein, AWS Organizations damit das Verwaltungskonto und das Zielkonto Zugriff auf die Protokolldaten gewähren können. CloudWatch

Auswahlkriterien für die Quelle

Stellen Sie sicher, dass die Quellauswahlkriterien Ihrer Zentralisierungsregel korrekt konfiguriert sind:

- **Konten und Regionen:** Stellen Sie sicher, dass die Quellkonten und Regionen, aus denen die Protokolle stammen, in der Regel enthalten sind. Protokollgruppen von Konten oder Regionen, die nicht in der Regel angegeben sind, werden nicht zentralisiert.
- **Protokollgruppenfilter:** Wenn Sie Protokollgruppenfilter konfiguriert haben, werden nur Protokollgruppen zentralisiert, die den angegebenen Kriterien entsprechen. Stellen Sie sicher, dass Ihre Auswahlkriterien für Protokollgruppen die Protokollgruppen enthalten, die Sie voraussichtlich zentralisieren werden.
- **Mitgliedschaft in einer Organisation:** Sowohl Quell- als auch Zielkonten müssen derselben AWS Organizations Organisation angehören. Konten außerhalb der Organisation können nicht an der Zentralisierung teilnehmen.

Kontingentlimit für Protokollgruppen erreicht

Wenn das Zielkonto das Kontingent für Protokollgruppen erreicht hat, können keine neuen Protokollgruppen für die Zentralisierung erstellt werden. Stellen Sie sicher, dass das Zielkonto

über ein ausreichendes Kontingent verfügt, um zentralisierte Protokollgruppen aus allen Quellkonten aufzunehmen. Sie können bei Bedarf eine Erhöhung des Kontingents beantragen.

Die Längenbeschränkung für den Namen des Protokollstreams wurde überschritten

Für Namen von Protokolldatenströmen gilt eine maximale Länge. Wenn die Zentralisierung Protokollstreams auf das Zielkonto repliziert, wird dem Namen des Protokolldatenstroms ein Suffix hinzugefügt. Wenn der resultierende Protokollstreamname die maximal zulässige Länge überschreitet, werden Datensätze gelöscht und es wird ein `InvalidLogStream` Fehler an das Kundenkonto gemeldet.

Gesundheitsstatus der Regel

Überprüfen Sie den Integritätsstatus der Zentralisierungsregel in der Konsole oder mithilfe der `GetCentralizationRuleForOrganization` API. Wenn die Regel als UNGESUND markiert ist, überprüfen Sie das `FailureReason` Feld auf spezifische Informationen zu dem Problem.

Um Zentralisierungsprobleme zu diagnostizieren, überprüfen Sie den Integritätsstatus der Zentralisierungsregel in der Konsole, überprüfen Sie die CloudWatch Metriken auf Fehler und Drosselung und untersuchen Sie die AWS CloudTrail Protokolle auf fehlgeschlagene API-Aufrufe. Weitere Informationen zu Zentralisierungsmetriken finden Sie unter. [Metriken und Dimensionen der Zentralisierung](#)

Arbeiten mit Protokollgruppen und Protokollstreams

Ein Protokollstream ist eine Abfolge von Protokollereignissen, die dieselbe Quelle nutzen. Jede einzelne Protokollquelle in CloudWatch Logs bildet einen separaten Protokollstream.

Eine Protokollgruppe ist eine Gruppe von Protokollstreams, die dieselben Einstellungen für die Aufbewahrung, Überwachung und Zugriffskontrolle besitzen. Sie können Protokollgruppen definieren und angeben, welche Streams in welche Gruppe geschickt werden sollen. Es gibt keine Begrenzung dazu, wie viele Protokollstreams zu einer Protokollgruppe gehören können.

Für Unternehmen, die Protokolldaten aus mehreren Konten und Regionen konsolidieren müssen, können Sie CloudWatch Logs Centralization verwenden, um Protokollgruppen automatisch auf ein zentrales Konto zu replizieren. Weitere Informationen finden Sie unter [Cross-account Regionsübergreifende Protokollzentralisierung](#).

Sie können die Verfahren in diesem Abschnitt verwenden, um mit Protokollgruppen und Protokollströmen zu arbeiten.

Erstellen Sie eine Protokollgruppe in CloudWatch Logs

Wenn Sie den CloudWatch Logs-Agenten mithilfe der Schritte in den vorherigen Abschnitten des Amazon CloudWatch Logs-Benutzerhandbuchs auf einer Amazon EC2-Instance installieren, wird die Protokollgruppe als Teil dieses Prozesses erstellt. Sie können eine Protokollgruppe auch direkt in der CloudWatch Konsole erstellen.

So erstellen Sie eine Protokollgruppe

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Log Management aus.
3. Wählen Sie Actions (Aktionen) und anschließend Create log group (Protokollgruppe erstellen) aus.
4. Geben Sie einen Namen für die Protokollgruppe ein. Wählen Sie anschließend Create log group (Protokollgruppe erstellen) aus.

 Tip

Sie können Protokoll-Gruppen sowie Dashboards und Alarme von der Menü Favorites and recents (Favoriten und Kürzliche) im Navigationsbereich als Favoriten festlegen. Bewegen Sie den Mauszeiger unter der Spalte Recently visited (Zuletzt besucht) über die Protokollgruppe, die Sie als Favorit markieren möchten, und wählen Sie das Sternsymbol daneben aus.

Protokolle an eine Protokollgruppe senden

CloudWatch Logs empfängt automatisch Protokollereignisse von verschiedenen AWS Diensten. Sie können auch andere Protokollereignisse mit einer der folgenden Methoden an CloudWatch Logs senden:

- CloudWatch Agent — Der vereinheitlichte CloudWatch Agent kann sowohl Metriken als auch CloudWatch Protokolle an Logs senden. Informationen zur Installation und Verwendung des CloudWatch Agenten finden Sie unter [Erfassung von Metriken und Protokollen von Amazon EC2 EC2-Instances und lokalen Servern mit dem CloudWatch Agenten im CloudWatch Amazon-Benutzerhandbuch](#).
- AWS CLI— [put-log-events](#)Lädt Stapel von Protokollereignissen in Logs hoch. CloudWatch
- Programmgesteuert — Mit der [PutLogEvents](#)API können Sie stapelweise Protokollereignisse programmgesteuert in Logs hochladen. CloudWatch

An Logs gesendete Protokolldaten anzeigen CloudWatch

Sie können Protokolldaten auf der stream-by-stream Grundlage anzeigen und durchblättern, wie sie vom CloudWatch Logs-Agenten an CloudWatch Logs gesendet wurden. Sie können den Zeitraum für die Anzeige der Protokolldaten festlegen.

So lassen Sie sich Protokolldaten anzeigen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Log Management aus.
3. Wählen Sie für Log Groups die Protokollgruppe, um die Streams anzuzeigen.

4. Wählen Sie in der Liste der Protokollgruppen den Namen der Protokollgruppe aus, die Sie anzeigen möchten.
5. Wählen Sie aus der Liste der Protokoll-Streams den Namen des Protokoll-Streams aus, den Sie anzeigen möchten.
6. Um die Ansicht der Protokolldaten zu ändern, führen Sie einen der folgenden Schritte aus:
 - Um ein einzelnes Protokollereignis zu erweitern, wählen Sie den Pfeil neben diesem Protokollereignis.
 - Um alle Protokollereignisse zu expandieren und diese als Nur-Text anzuzeigen, wählen Sie über der obigen Liste der Protokollereignisse die Option Text.
 - Um die Protokollereignisse zu filtern, geben Sie den gewünschten Suchfilter in das Suchfeld ein. Weitere Informationen finden Sie unter [Erstellen von Metriken aus Protokollereignissen mithilfe von Filtern](#).
 - Um Protokolldaten für einen bestimmten Datums- und Zeitbereich anzuzeigen, wählen Sie neben dem Suchfilter den Pfeil neben Datum und Uhrzeit aus. Um einen Datums- und Zeitbereich festzulegen, wählen Sie Absolute (Absolut) aus. Um eine vordefinierte Anzahl von Minuten, Stunden, Tagen oder Wochen auszuwählen, wählen Sie Relative (Relativ) aus. Sie können auch zwischen UTC und lokaler Zeitzone wechseln.

Suchen von Protokolldaten mithilfe von Filtermustern

Sie können Ihre Protokolldaten mithilfe der [Filtermustersyntax für Metrikfilter, Abonnementfilter, Filterprotokollereignisse und Live Tail](#) suchen. Sie können alle Protokolldatenströme innerhalb einer Protokollgruppe durchsuchen, oder AWS CLI Sie können mithilfe von auch nach bestimmten Protokolldatenströmen suchen. Wenn die Suche ausgeführt wird, kehrt sie zur ersten Daten der gefundenen Daten und einem Token zurück, um die nächste Datenseite aufzurufen oder die Suche fortzusetzen. Wenn keine Ergebnisse zurückgegeben werden, können Sie die Suche fortsetzen.

Sie können den Zeitbereich einschränken, den Sie abfragen möchten, und damit den Umfang der Suche reduzieren. Sie können mit einem größeren Bereich beginnen, um zu erkennen, ob die Protokollzeilen, die Sie interessieren, dazu gehören. Dann können Sie den Zeitraum auf diesen Umfang verkürzen, um die Protokolle für den Zeitbereich anzuzeigen, der für Sie von Interesse ist.

Sie können auch direkt von den aus den Protokollen extrahierten Metriken zu den entsprechenden Protokolle wechseln.

Wenn Sie mit einem Konto angemeldet sind, das als Überwachungskonto in CloudWatch kontenübergreifender Observability eingerichtet wurde, können Sie Protokollereignisse der Quellkonten, die mit diesem Überwachungskonto verknüpft sind, suchen und filtern. Weitere Informationen finden Sie unter [Kontoübergreifende Beobachtbarkeit von CloudWatch](#).

Suchen von Protokolleinträge mithilfe der Konsole

Sie können mithilfe der Konsole nach den Protokolleinträgen suchen, die ein bestimmtes Kriterien erfüllen.

So suchen Sie Ihre Protokolle mithilfe der Konsole

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im Navigationsbereich Log Management aus.
3. Wählen Sie für Log Groups den Namen der Protokollgruppe mit dem Protokoll-Stream aus, nach dem gesucht werden soll.
4. Wählen Sie für Log Streams den Namen des zu suchenden Protokoll-Streams.
5. Geben Sie unter Protokollereignisse die zu verwendende Filtersyntax ein.

So suchen Sie alle Protokolleinträge für einen Zeitbereich mithilfe der Konsole

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Log Management aus.
3. Wählen Sie für Log Groups den Namen der Protokollgruppe mit dem Protokoll-Stream aus, nach dem gesucht werden soll.
4. Wählen Sie Protokollgruppe suchen aus.
5. Wählen Sie für Protokollereignisse den Datums- und Uhrzeitbereich aus und geben Sie die Filtersyntax ein.

Suchen Sie nach Protokolleinträgen mithilfe der AWS CLI

Sie können mit dem nach Protokolleinträgen suchen, die ein bestimmtes Kriterium erfüllen AWS CLI.

Um Protokolleinträge mit dem zu suchen AWS CLI

Führen Sie an einer Eingabeaufforderung den folgenden [filter-log-events](#)-Befehl aus. Mit `--filter-pattern` begrenzen Sie die Ergebnisse auf das angegebene Filtermuster, und mit `--log-stream-names` begrenzen Sie die Ergebnisse auf die angegebene Protokollstreams.

```
aws logs filter-log-events --log-group-name my-group [--log-stream-names LIST_OF_STREAMS_TO_SEARCH] [--filter-pattern VALID_METRIC_FILTER_PATTERN]
```

Um Protokolleinträge über einen bestimmten Zeitraum zu durchsuchen, verwenden Sie AWS CLI

Führen Sie in einer Befehlszeile den folgenden [filter-log-events](#) Befehl aus:

```
aws logs filter-log-events --log-group-name my-group [--log-stream-names LIST_OF_STREAMS_TO_SEARCH] [--start-time 1482197400000] [--end-time 1482217558365] [--filter-pattern VALID_METRIC_FILTER_PATTERN]
```

Wechseln von Metriken zu Protokollen

Sie können aus anderen Teilen der Konsole zu bestimmten Protokolleinträge wechseln.

So rufen Sie Protokolle von Dashboard-Widgets aus auf

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Dashboards aus.
3. Wählen Sie ein Dashboard.
4. Wählen Sie im Widget das Symbol zum Anzeigen der Protokolle und dann die Option View logs in this time range. Wenn mehrere Metrikfilter vorhanden sind, wählen Sie einen Filter aus der Liste aus. Wenn mehr Metrikfilter vorhanden sind, als in der Liste angezeigt werden, wählen Sie More metric filters (Weitere Metrikfilter) aus und wählen Sie einen Metrikfilter aus oder suchen Sie danach.

So rufen Sie Protokolle aus Metriken ab

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Metriken aus.
3. Geben Sie in das Suchfeld auf der Registerkarte All metrics den Namen der Metrik ein und drücken Sie die Eingabetaste.
4. Wählen Sie eine oder mehrere Metriken aus den Suchergebnissen aus.

5. Wählen Sie Actions und View logs aus. Wenn mehrere Metrikfilter vorhanden sind, wählen Sie einen Filter aus der Liste aus. Wenn mehr Metrikfilter vorhanden sind, als in der Liste angezeigt werden, wählen Sie More metric filters (Weitere Metrikfilter) aus und wählen Sie einen Metrikfilter aus oder suchen Sie danach.

Fehlerbehebung

Suche dauert zu lange

Wenn viele Protokolldaten vorhanden sind, kann die Suche sehr lange dauern. Gehen Sie wie folgt vor, um die Suche zu beschleunigen:

- Wenn Sie den verwenden AWS CLI, können Sie die Suche auf die Protokollstreams beschränken, an denen Sie interessiert sind. Wenn Ihre Protokollgruppe beispielsweise 1000 Logstreams hat, Sie aber nur drei Logstreams sehen möchten, von denen Sie wissen, dass sie relevant sind, können Sie die verwenden, AWS CLI um Ihre Suche auf diese drei Logstreams innerhalb der Protokollgruppe zu beschränken.
- Verwenden Sie einen kürzeren, individuelleren Zeitraum, wodurch die Menge der durchsuchten Daten reduziert und die Abfrage beschleunigt werden.

Ändern Sie die Aufbewahrung von Protokolldaten in CloudWatch Logs

Standardmäßig werden Protokolldaten auf unbestimmte Zeit in CloudWatch Logs gespeichert. Sie können jedoch konfigurieren, wie lange Protokolldaten in einer Protokollgruppe gespeichert werden sollen. Alle Daten, die älter sind als die aktuelle Aufbewahrungseinstellung, werden gelöscht. Sie können die Protokoll-Aufbewahrung für jede Protokollgruppe jederzeit ändern.

Note

CloudWatch Logs löscht Protokollereignisse nicht sofort, wenn sie ihre Aufbewahrungseinstellung erreichen. In der Regel dauert es danach bis zu 72 Stunden, bis Protokollereignisse gelöscht werden. In seltenen Fällen kann es jedoch länger dauern. Das heißt, wenn Sie eine Protokollgruppe so ändern, dass sie eine längere Aufbewahrungseinstellung hat, wenn sie Protokollereignisse enthält, die das Ablaufdatum überschritten haben, aber noch nicht gelöscht wurden, dauert es bis zu 72 Stunden, bis diese

Protokollereignisse gelöscht werden, nachdem das neue Aufbewahrungsdatum erreicht wurde. Um sicherzustellen, dass Protokolldaten dauerhaft gelöscht werden, behalten Sie eine Protokollgruppe bei ihrer niedrigeren Aufbewahrungseinstellung, bis 72 Stunden nach Ablauf der vorherigen Aufbewahrungsfrist vergangen sind oder Sie bestätigt haben, dass die älteren Protokollereignisse gelöscht wurden.

Wenn Protokollereignisse ihre Aufbewahrungseinstellung erreichen, werden sie zum Löschen markiert. Nachdem sie zum Löschen markiert wurden, erhöhen sie Ihre Archivierungskosten nicht mehr, auch wenn sie erst zu einem späteren Zeitpunkt tatsächlich gelöscht werden.

Diese zum Löschen markierten Protokollereignisse sind ebenfalls nicht enthalten, wenn Sie eine API zum Abrufen des `storedBytes`-Werts verwenden, um zu sehen, wie viele Byte eine Protokollgruppe speichert.

So ändern Sie die Einstellungen für die Aufbewahrung der Protokolldateien

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle), Log groups (Protokollgruppen) aus.
3. Gehen Sie zur Protokollgruppe, die aktualisiert werden soll.
4. Wählen Sie in der Spalte Aufbewahrung für diese Protokollgruppe die aktuelle Aufbewahrungseinstellung aus, z. B. Niemals ablaufen.
5. Wählen Sie unter Aufbewahrungseinstellung für Ereignisse ablaufen danach einen Wert für die Protokollaufbewahrung aus, und klicken Sie dann auf Speichern.

Protokollgruppen vor dem Löschen schützen

Sie können optional den Löschschutz aktivieren, um ein versehentliches Löschen wichtiger Protokollgruppen zu verhindern. Ausführliche Informationen zum Löschschutz finden Sie unter [Protokollgruppen vor dem Löschen schützen](#).

Protokollgruppen vor dem Löschen schützen

Löschschutz aktivieren

Sie können den Löschschutz aktivieren, wenn Sie eine neue Protokollgruppe oder für bestehende Protokollgruppen erstellen. Wählen Sie bei der Erstellung der Protokollgruppe „Löschschutz aktiviert“

oder übergeben Sie den Parameter `--deletion-protection-enabled`. Standardmäßig ist der Löschschutz nicht aktiviert.

Um den Löschschutz für eine bestehende Protokollgruppe (Konsole) zu aktivieren oder zu deaktivieren

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Log Management aus.
3. Wählen Sie die Protokollgruppe aus, die Sie schützen möchten.
4. Wählen Sie Aktionen, Löschschutz bearbeiten.
5. Überprüfen Sie im Dialogfeld die Änderungen und reichen Sie sie anschließend ein.

Wenn Sie den verwenden AWS CLI, um den Löschschutz für eine bestehende Protokollgruppe zu aktivieren:

```
aws logs put-log-group-deletion-protection \  
--log-group-identifizier "/my-application/logs" \  
--deletion-protection-enabled
```

Um den Löschschutz für eine bestehende Protokollgruppe zu entfernen:

```
aws logs put-log-group-deletion-protection \  
--log-group-identifizier "/my-application/logs" \  
--no-deletion-protection-enabled
```

Fehlerbehandlung

Wenn Sie versuchen, eine Protokollgruppe mit aktiviertem Löschschutz zu löschen, erhalten Sie folgende Meldung: „Die Protokollgruppe kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist. `ValidationException` Deaktivieren Sie zuerst den Löschschutz.“

Taggen Sie Protokollgruppen in Amazon CloudWatch Logs

Sie können den Protokollgruppen, die Sie in Amazon CloudWatch Logs erstellen, Ihre eigenen Metadaten in Form von Tags zuweisen. Ein Tag ist ein Schlüssel-Wert-Paar, das Sie für eine Protokollgruppe definieren. Die Verwendung von Tags ist eine einfache und dennoch leistungsstarke Möglichkeit, AWS Ressourcen zu verwalten und Daten, einschließlich Rechnungsdaten, zu organisieren.

 Note

Sie können Tags verwenden, um den Zugriff auf CloudWatch Logs-Ressourcen, einschließlich Protokollgruppen und -ziele, zu kontrollieren. Der Zugriff auf Protokollstreams wird aufgrund der hierarchischen Beziehung zwischen Protokollgruppen und Protokollstreams auf der Ebene der Protokollgruppen gesteuert. Informationen über die Verwendung von Tags zum Steuern des Zugriffs finden Sie unter [Steuern des Zugriffs auf Amazon-Web-Services-Ressourcen mithilfe von Tags](#).

Inhalt

- [Grundlagen zu Tags \(Markierungen\)](#)
- [Verfolgen der Kosten mithilfe von Markierungen](#)
- [Tag-Einschränkungen](#)
- [Taggen von Protokollgruppen mit dem AWS CLI](#)
- [Taggen von Protokollgruppen mithilfe der CloudWatch Logs-API](#)

Grundlagen zu Tags (Markierungen)

Sie verwenden AWS CloudFormation die oder CloudWatch Logs API AWS CLI, um die folgenden Aufgaben auszuführen:

- Hinzufügen von Tags zu einer Protokollgruppe während der Erstellung
- Hinzufügen von Tags zu einer vorhandenen Protokollgruppe
- Auflistung der Tags für eine Protokollgruppe
- Entfernung der Tags von einer Protokollgruppe

Sie können Tags verwenden, um Ihre Protokollgruppen zu kategorisieren. Sie können sie beispielsweise nach Zweck, Inhaber oder Umgebung kategorisieren. Da Sie für jeden Tag den Schlüssel und Wert definieren, können Sie eine auf benutzerdefinierte Reihe von Kategorien anlegen, die Ihren jeweiligen Anforderungen gerecht wird. Sie könnten zum Beispiel eine Reihe von Tags definieren, mit der Sie Protokollgruppen nach Inhaber und zugehöriger Anwendung nachverfolgen können. Im Folgenden finden Sie einige Beispiele für Tags:

- Projekt: Projektname

- Inhaber: Name
- Zweck: Belastungstest
- Anwendung: Anwendungsname
- Umgebung: Produktion

Verfolgen der Kosten mithilfe von Markierungen

Sie können Tags verwenden, um Ihre AWS Kosten zu kategorisieren und nachzuverfolgen. Wenn Sie Tags auf Ihre AWS Ressourcen, einschließlich Protokollgruppen, anwenden, enthält Ihr AWS Kostenzuordnungsbericht die Nutzung und die Kosten, die nach Stichwörtern zusammengefasst sind. Sie können Tags anwenden, die geschäftliche Kategorien (wie Kostenstellen, Anwendungsnamen oder Eigentümer) darstellen, um die Kosten für mehrere Services zu organisieren. Weitere Informationen finden Sie unter [Verwenden von Kostenzuordnungs-Tags für benutzerdefinierte Fakturierungsberichte](#) im AWS Billing -Benutzerhandbuch.

Tag-Einschränkungen

Für Tags gelten die folgenden Einschränkungen.

Grundlegende Einschränkungen

- Die maximale Anzahl von Tags pro Protokollgruppe beträgt 50.
- Bei Tag-Schlüsseln und -Werten muss die Groß- und Kleinschreibung beachtet werden.
- Sie können Tags für eine gelöschte Protokollgruppe nicht ändern oder bearbeiten.

Einschränkungen für Tag-Schlüssel

- Jeder Tag-Schlüssel muss einmalig sein. Wenn Sie einen Tag mit einem Schlüssel hinzufügen, der bereits verwendet wird, wird das vorhandene Schlüssel-Wert-Paar durch den neuen Tag überschrieben.
- Sie können mit einem Tag-Schlüssel nicht beginnen `aws:`, da dieses Präfix für die Verwendung durch AWS reserviert ist. AWS erstellt in Ihrem Namen Tags, die mit diesem Präfix beginnen, aber Sie können sie nicht bearbeiten oder löschen.
- Tag-Schlüssel müssen zwischen 1 und 128 Unicode-Zeichen lang sein.
- Tag-Schlüssel müssen die folgenden Zeichen enthalten: Unicode-Zeichen, Ziffern, Leerzeichen sowie die folgenden Sonderzeichen: `_ . / = + - @`.

Einschränkungen für den Tag-Wert

- Tag-Werte müssen zwischen 0 und 255 Unicode-Zeichen lang sein.
- Tag-Werte können leer sein. Ansonsten müssen sie die folgenden Zeichen enthalten: Unicode-Zeichen, Ziffern, Leerzeichen und eines der folgenden Sonderzeichen: `_ . / = + - @`.

Taggen von Protokollgruppen mit dem AWS CLI

Sie können Tags mithilfe der AWS CLI hinzufügen, auflisten und entfernen. Beispiele finden Sie in der folgenden Dokumentation:

[create-log-group](#)

Erstellt eine Protokollgruppe. Sie können beim Erstellen einer Protokollgruppe optional Tags hinzufügen.

[tag-resource](#)

Weist der angegebenen Logs-Ressource ein oder mehrere Tags (Schlüssel-Wert-Paare) zu. CloudWatch

[list-tags-for-resource](#)

Zeigt die Tags an, die einer CloudWatch Logs-Ressource zugeordnet sind.

[untag-resource](#)

Entfernt ein oder mehrere Tags aus der angegebenen CloudWatch Logs-Ressource.

Taggen von Protokollgruppen mithilfe der CloudWatch Logs-API

Mithilfe der CloudWatch Logs-API können Sie Tags hinzufügen, auflisten und entfernen. Beispiele finden Sie in der folgenden Dokumentation:

[CreateLogGroup](#)

Erstellt eine Protokollgruppe. Sie können beim Erstellen einer Protokollgruppe optional Tags hinzufügen.

[TagResource](#)

Weist der angegebenen CloudWatch Logs-Ressource ein oder mehrere Tags (Schlüssel-Wert-Paare) zu.

[ListTagsForResource](#)

Zeigt die Tags an, die einer CloudWatch Logs-Ressource zugeordnet sind.

[UntagResource](#)

Entfernt ein oder mehrere Tags aus der angegebenen CloudWatch Logs-Ressource.

Verschlüsseln Sie Protokolldaten in CloudWatch Logs mit AWS Key Management Service

Protokollgruppendaten werden in CloudWatch Logs immer verschlüsselt. Standardmäßig verwendet CloudWatch Logs serverseitige Verschlüsselung mit 256-Bit-AES-GCM (Advanced Encryption Standard Galois/Counter Mode), um Protokolldaten im Ruhezustand zu verschlüsseln. Als Alternative können Sie AWS Key Management Service für diese Verschlüsselung verwenden. Wenn Sie dies tun, erfolgt die Verschlüsselung mithilfe eines Schlüssels. AWS KMS Die Verschlüsselung mit AWS KMS wird auf Protokollgruppenebene aktiviert, indem ein KMS-Schlüssel einer Protokollgruppe zugeordnet wird, entweder wenn Sie die Protokollgruppe erstellen oder nachdem sie existiert.

Important

CloudWatch Logs unterstützt jetzt den Verschlüsselungskontext, indem er `kms:EncryptionContext:aws:logs:arn` als Schlüssel und den ARN der Protokollgruppe als Wert für diesen Schlüssel verwendet. Wenn es Protokollgruppen gibt, die Sie bereits mit einem KMS-Schlüssel verschlüsselt haben, und Sie die Verwendung des Schlüssels auf ein einzelnes Konto und eine einzelne Protokollgruppe beschränken möchten, sollten Sie einen neuen KMS-Schlüssel zuweisen, der eine Bedingung in der IAM-Richtlinie enthält. Weitere Informationen finden Sie unter [AWS KMS Schlüssel und Verschlüsselungskontext](#).

Important

CloudWatch Logs unterstützt jetzt `kms:ViaService`, sodass Logs in Ihrem Namen AWS KMS Anrufe tätigen können. Sie sollten dies zu Ihren Rollen hinzufügen, die CloudWatch Logs entweder in Ihrer Schlüsselrichtlinie oder in IAM aufrufen. Weitere Informationen finden Sie unter [kms: ViaService](#).

Nachdem Sie einen KMS-Schlüssel mit einer Protokollgruppe verknüpft haben, werden alle für die Protokollgruppe neu übernommenen Daten mithilfe des Schlüssels verschlüsselt. Diese Daten werden während ihrer gesamten Aufbewahrungsfrist in verschlüsseltem Format gespeichert. CloudWatch Logs entschlüsselt diese Daten, wann immer sie angefordert werden. CloudWatch Protokolle müssen über Berechtigungen für den KMS-Schlüssel verfügen, wenn verschlüsselte Daten angefordert werden.

Wenn Sie später die Zuordnung eines KMS-Schlüssels zu einer Protokollgruppe aufheben, verschlüsselt CloudWatch Logs neu aufgenommene Daten mit der Standardverschlüsselungsmethode CloudWatch Logs. Alle zuvor aufgenommenen Daten, die mit dem KMS-Schlüssel verschlüsselt wurden, bleiben mit dem KMS-Schlüssel verschlüsselt. CloudWatch Protokolle können diese Daten auch dann zurückgeben, wenn die Zuordnung des KMS-Schlüssels aufgehoben wurde, da CloudWatch Protokolle weiterhin auf den Schlüssel verweisen können. Wenn der Schlüssel jedoch später deaktiviert wird, kann Logs die CloudWatch Protokolle, die mit diesem Schlüssel verschlüsselt wurden, nicht lesen.

Important

CloudWatch Logs unterstützt nur symmetrische KMS-Schlüssel. Verwenden Sie keinen asymmetrischen Schlüssel, um die Daten in Ihren Protokollgruppen zu verschlüsseln.

Weitere Informationen finden Sie unter [Verwenden von symmetrischen und asymmetrischen Schlüsseln](#).

Einschränkungen

- Um die folgenden Schritte ausführen zu können, benötigen Sie die folgenden Berechtigungen: `kms:CreateKey`, `kms:GetKeyPolicy` und `kms:PutKeyPolicy`.
- Nachdem Sie die Verknüpfung eines Schlüssels mit einer Protokollgruppe hergestellt oder aufgehoben haben, kann es bis zu fünf Minuten dauern, bis der Vorgang wirksam wird.
- Wenn Sie CloudWatch Logs den Zugriff auf einen zugehörigen Schlüssel entziehen oder einen zugehörigen KMS-Schlüssel löschen, können Ihre verschlüsselten Daten in CloudWatch Logs nicht mehr abgerufen werden.
- Sie können einen KMS-Schlüssel nicht mithilfe der CloudWatch Konsole einer vorhandenen Protokollgruppe zuordnen.

Schritt 1: Erstellen Sie einen AWS KMS Schlüssel

Um einen KMS-Schlüssel zu erstellen, verwenden Sie den folgenden Befehl [create-key](#):

```
aws kms create-key
```

Die Ausgabe enthält die Schlüssel-ID und den Amazon-Ressourcennamen (ARN) des Schlüssels. Das Folgende ist eine Beispielausgabe:

```
{
  "KeyMetadata": {
    "Origin": "AWS_KMS",
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "Description": "",
    "KeyManager": "CUSTOMER",
    "Enabled": true,
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "KeyState": "Enabled",
    "CreationDate": 1478910250.94,
    "Arn": "arn:aws:kms:us-west-2:123456789012:key/6f815f63-e628-448c-8251-
e40cb0d29f59",
    "AWSAccountId": "123456789012",
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ]
  }
}
```

Schritt 2: Festlegen von Berechtigungen auf dem KMS-Schlüssel

Standardmäßig sind alle AWS KMS Schlüssel privat. Nur der Ressourcenbesitzer kann mit ihnen Daten verschlüsseln und entschlüsseln. Der Ressourceninhaber kann jedoch anderen Benutzern und Ressourcen Zugriffsberechtigungen für den KMS-Schlüssel erteilen. Mit diesem Schritt erteilen Sie dem CloudWatch Logs-Dienstprinzipal und der Anruferrolle die Erlaubnis, den Schlüssel zu verwenden. Dieser Dienstprinzipal muss sich in derselben AWS Region befinden, in der der KMS-Schlüssel gespeichert ist.

Als bewährte Methode empfehlen wir, die Verwendung des KMS-Schlüssels auf die von Ihnen angegebenen AWS Konten oder Protokollgruppen zu beschränken.

Speichern Sie zunächst die Standardrichtlinie für Ihren KMS-Schlüssel `policy.json` mit dem folgenden [get-key-policy](#) Befehl:

```
aws kms get-key-policy --key-id key-id --policy-name default --output text > ./  
policy.json
```

Öffnen Sie die Datei `policy.json` in einem Texteditor und fügen Sie den in Fettschrift angezeigten Abschnitt aus einer der folgenden Anweisungen hinzu. Sie trennen die vorhandene Anweisung von der neuen Anweisung durch ein Komma. In diesen Anweisungen `Condition` werden Abschnitte verwendet, um die Sicherheit des AWS KMS Schlüssels zu erhöhen. Weitere Informationen finden Sie unter [AWS KMS Schlüssel und Verschlüsselungskontext](#).

Der `Condition`-Abschnitt in diesem Beispiel schränkt den Schlüssel auf einen einzelnen Protokollgruppen-ARN ein.

JSON

```
{  
  "Version": "2012-10-17",  
  "Id": "key-default-1",  
  "Statement": [  
    {  
      "Sid": "Enable IAM User Permissions",  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::123456789012:root"  
      },  
      "Action": "kms:*",  
      "Resource": "*"   
    },  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "Service": "logs.us-east-1.amazonaws.com"  
      },  
      "Action": [  
        "kms:Encrypt",  
        "kms:Decrypt",  
        "kms:ReEncrypt*",  
        "kms:GenerateDataKey*",  
        "kms:Describe"
```

```

    ],
    "Resource": "*",
    "Condition": {
        "ArnEquals": {
            "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:111122223333:log-group:log-group-name"
        }
    }
}

```

Der Condition-Abschnitt in diesem Beispiel beschränkt die Verwendung des AWS KMS - Schlüssels auf das angegebene Konto, kann jedoch für jede Protokollgruppe verwendet werden.

JSON

```

{
  "Version": "2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:Describe*"
      ]
    }
  ]
}

```

```

    ],
    "Resource": "*",
    "Condition": {
      "ArnLike": {
        "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:123456789012:"
      }
    }
  }
]
}

```

Schritt 3: Legen Sie die Berechtigungen für den aufrufenden IAM-Prinzipal fest

Fügen Sie dem IAM-Prinzipal (Benutzer oder Rolle), der Logs aufrufen CloudWatch soll, Berechtigungen hinzu. APIs Die erforderlichen Berechtigungen hängen davon ab, welche Operationen der Principal ausführen muss. Sie können diese Berechtigungen in der AWS KMS Schlüsselrichtlinie oder über IAM für die Rolle selbst hinzufügen. CloudWatch Logs verwendet `kms:ViaService`, um AWS KMS Anrufe im Namen des Kunden zu tätigen. Weitere Informationen finden Sie unter [kms: ViaService](#).

Berechtigungen für die Zuordnung eines KMS-Schlüssels zu einer Protokollgruppe

Der IAM-Prinzipal, der `CreateLogGroup` mit einem oder `kmsKeyId` mehreren Parametern aufruft `AssociateKmsKey`, muss über `kms:DescribeKey` Berechtigungen für den angegebenen KMS-Schlüssel verfügen. Wenn der Aufrufer nicht über diese Berechtigung verfügt, schlägt der API-Aufruf mit einer fehl. `AccessDeniedException`

Das folgende Beispiel für eine Schlüsselrichtlinienanweisung gewährt die Mindestberechtigungen, die erforderlich sind, um einen KMS-Schlüssel einer Protokollgruppe zuzuordnen:

```

{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::account_id:role/role_name"
  },
  "Action": [
    "kms:Describe*"
  ],

```

```

"Resource": "*",
"Condition": {
  "StringEquals": {
    "kms:ViaService": [
      "logs.region.amazonaws.com"
    ]
  }
}
}

```

Berechtigungen zum Lesen und Schreiben verschlüsselter Protokolldaten

Wenn der IAM-Prinzipal auch verschlüsselte Protokolldaten lesen oder schreiben muss (z. B. beim Aufrufen von `PutLogEvents`, `GetLogEvents`, `FilterLogEvents`, oder `StartQuery` in einer Protokollgruppe, die mit einem vom Kunden verwalteten KMS-Schlüssel verschlüsselt ist), benötigt der Principal zusätzliche AWS KMS Berechtigungen. Das folgende Beispiel für eine wichtige Richtlinienanweisung gewährt den vollständigen Satz von Berechtigungen, die sowohl für die Zuordnung eines Schlüssels als auch für das Lesen oder Schreiben verschlüsselter Protokolldaten erforderlich sind:

```

{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::account_id:role/role_name"
  },
  "Action": [
    "kms:Encrypt",
    "kms:ReEncrypt*",
    "kms:Decrypt",
    "kms:GenerateDataKey*",
    "kms:Describe*"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": [
        "logs.region.amazonaws.com"
      ]
    }
  }
}

```

Es hat sich bewährt, die Richtlinie nur auf die Rollen auszudehnen, die mit AWS KMS verschlüsselten Protokollgruppen interagieren.

Alternativ können Sie der Anruferrolle die folgende Richtlinie hinzufügen, um alle Berechtigungen sowohl für die Zuordnung eines Schlüssels als auch für das Lesen oder Schreiben verschlüsselter Protokolldaten über IAM zu gewähren. Dies kann einer vorhandenen Rollenrichtlinie hinzugefügt oder einer Rolle als zusätzliche separate Richtlinie zugewiesen werden. Wenn Sie diese Methode verwenden, sollten Sie als bewährte Methode die Richtlinie nur auf die AWS KMS Schlüssel beschränken, die für die Protokollverschlüsselung verwendet werden. Weitere Informationen finden Sie unter [IAM-Richtlinien bearbeiten](#).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:ReEncrypt*",
        "kms:Decrypt",
        "kms:GenerateDataKey*",
        "kms:Describe*"
      ],
      "Condition": {
        "StringEquals": {
          "kms:ViaService": [
            "logs.us-east-1.amazonaws.com"
          ]
        }
      },
      "Resource": "arn:aws:kms:us-east-1:444455556666:key/key_id"
    }
  ]
}
```

Fügen Sie abschließend die aktualisierte Richtlinie mit dem folgenden [put-key-policy](#) Befehl hinzu:

```
aws kms put-key-policy --key-id key-id --policy-name default --policy file://  
policy.json
```

Schritt 4: Ordnen Sie einen KMS-Schlüssel einer Protokollgruppe zu

Sie können einen KMS-Schlüssel während oder nach der Erstellung einer Protokollgruppe mit dieser verknüpfen.

Verwenden Sie den folgenden [describe-log-groups](#) Befehl, um herauszufinden, ob einer Protokollgruppe bereits ein KMS-Schlüssel zugeordnet ist:

```
aws logs describe-log-groups --log-group-name-prefix "log-group-name-prefix"
```

Wenn die Ausgabe ein `kmsKeyId`-Feld enthält, wird die Protokollgruppe dem Schlüssel zugeordnet, der für den Wert dieses Feldes angezeigt wird.

So verknüpfen Sie einen KMS-Schlüssel mit einer Protokollgruppe, wenn Sie sie erstellen

Verwenden Sie den [create-log-group](#) Befehl wie folgt:

```
aws logs create-log-group --log-group-name my-log-group --kms-key-id "key-arn"
```

So verknüpfen Sie den KMS-Schlüssel mit einer vorhandenen Protokollgruppe

Verwenden Sie den [associate-kms-key](#) Befehl wie folgt:

```
aws logs associate-kms-key --log-group-name my-log-group --kms-key-id "key-arn"
```

Schritt 5: Trennen Sie die Zuordnung des Schlüssels zu einer Protokollgruppe

Verwenden Sie den folgenden [disassociate-kms-key](#) Befehl, um die Zuordnung des KMS-Schlüssels, der einer Protokollgruppe zugeordnet ist, aufzuheben:

```
aws logs disassociate-kms-key --log-group-name my-log-group
```

AWS KMS Schlüssel und Verschlüsselungskontext

Um die Sicherheit Ihrer AWS Key Management Service Schlüssel und Ihrer verschlüsselten Protokollgruppen zu erhöhen, fügt CloudWatch Logs die Protokollgruppe jetzt ARNs als Teil des Verschlüsselungskontextes hinzu, der zur Verschlüsselung Ihrer Protokolldaten verwendet wird. Der Verschlüsselungskontext ist ein Satz von Schlüssel-Wert-Paaren, die als zusätzliche authentifizierte Daten verwendet werden. Der Verschlüsselungskontext ermöglicht es Ihnen, mithilfe der IAM-Richtlinienbedingungen den Zugriff auf Ihren AWS KMS Schlüssel nach AWS Konto und Protokollgruppe zu beschränken. Weitere Informationen finden Sie unter [Verschlüsselungskontext](#) und [IAM-JSON-Richtlinienelemente: Bedingung](#).

Sie sollten für jede verschlüsselte Protokollgruppe einen anderen KMS-Schlüssel verwenden.

Wenn Sie zuvor eine Protokollgruppe verschlüsselt haben und jetzt für diese Protokollgruppe einen neuen KMS-Schlüssel verwenden möchten, der nur für diese Protokollgruppe funktioniert, gehen Sie folgendermaßen vor.

So konvertieren Sie eine verschlüsselte Protokollgruppe für die Verwendung eines KMS-Schlüssels mit einer Richtlinie, die diesen auf diese Protokollgruppe einschränkt

1. Geben Sie den folgenden Befehl ein, um den ARN des aktuellen Schlüssels der Protokollgruppe zu finden:

```
aws logs describe-log-groups
```

Die Ausgabe enthält die folgende Zeile. Notieren Sie den ARN. Sie benötigen ihn in Schritt 7.

```
...  
"kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/01234567-89ab-  
cdef-0123-456789abcdef"  
...
```

2. Geben Sie den folgenden Befehl ein, um einen neuen KMS-Schlüssel zu erstellen:

```
aws kms create-key
```

3. Geben Sie den folgenden Befehl ein, um die Richtlinie des neuen Schlüssels in einer `policy.json`-Datei zu speichern:

```
aws kms get-key-policy --key-id new-key-id --policy-name default --output text > ./policy.json
```

4. Verwenden Sie einen Texteditor zum Öffnen von `policy.json` und fügen Sie der Richtlinie einen Condition-Ausdruck hinzu:

JSON

```
{
  "Version": "2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:Describe*"
      ],
      "Resource": "*",
      "Condition": {
        "ArnLike": {
          "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:111122223333:log-group:LOG-GROUP-NAME"
        }
      }
    }
  ]
}
```



```
]
}
```

5. Geben Sie den folgenden Befehl ein, um dem neuen KMS-Schlüssel die aktualisierte Richtlinie hinzuzufügen:

```
aws kms put-key-policy --key-id new-key-ARN --policy-name default --policy file://policy.json
```

6. Geben Sie den folgenden Befehl ein, um die Richtlinie mit Ihrer Protokollgruppe zu verknüpfen:

```
aws logs associate-kms-key --log-group-name my-log-group --kms-key-id new-key-ARN
```

CloudWatch Logs verschlüsselt jetzt alle neuen Daten mit dem neuen Schlüssel.

7. Als Nächstes widerrufen Sie alle Berechtigungen außer Decrypt aus dem alten Schlüssel. Geben Sie zunächst den folgenden Befehl ein, um die alte Richtlinie abzurufen:

```
aws kms get-key-policy --key-id old-key-ARN --policy-name default --output text > ./policy.json
```

8. Verwenden Sie einen Texteditor, um Action zu öffnen und alle Werte aus der Liste `policy.json` außer `kms:Decrypt` zu entfernen.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "logs.region.amazonaws.com"
      },
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "*"
    }
  ]
}
```

9. Geben Sie den folgenden Befehl ein, um dem alten Schlüssel die aktualisierte Richtlinie hinzuzufügen:

```
aws kms put-key-policy --key-id old-key-ARN --policy-name default --policy file://policy.json
```

Den Schutz vertraulicher Protokolldaten mit Maskierung unterstützen

Mithilfe von Datenschutzrichtlinien für Protokollgruppen können Sie dazu beitragen, sensible Daten, die in CloudWatch Logs aufgenommen werden, zu schützen. Mit diesen Richtlinien können Sie sensible Daten prüfen und maskieren, die in Protokollereignissen vorkommen, die von den Protokollgruppen in Ihrem Konto erfasst werden.

Wenn Sie eine Datenschutzrichtlinie erstellen, werden vertrauliche Daten, die den von Ihnen ausgewählten Datenkennungen entsprechen, standardmäßig an allen Ausgangspunkten maskiert, einschließlich CloudWatch Logs Insights, Metrikfiltern und Abonnementfiltern. Nur Benutzer mit der IAM-Berechtigung `logs:Unmask` können unmaskierte Daten anzeigen.

Sie können eine Datenschutzrichtlinie für alle Protokollgruppen in Ihrem Konto erstellen, und Sie können auch eine Datenschutzrichtlinie für einzelne Protokollgruppen erstellen. Wenn Sie eine Richtlinie für Ihr gesamtes Konto erstellen, gilt sie sowohl für bestehende Protokollgruppen als auch für Protokollgruppen, die in Zukunft erstellt werden.

Wenn Sie eine Datenschutzrichtlinie für Ihr gesamtes Konto und gleichzeitig eine Richtlinie für eine einzelne Protokollgruppe erstellen, gelten beide Richtlinien für diese Protokollgruppe. Alle verwalteten Datenidentifikatoren, die in einer der beiden Richtlinien angegeben sind, werden in dieser Protokollgruppe geprüft und maskiert.

 Note

Das Maskieren sensibler Daten wird für Protokollgruppen sowohl in den Protokollklassen Standard als auch in der Protokollklasse für seltenen Zugriff unterstützt. Weitere Hinweise zu Protokollklassen finden Sie unter. [Klassen protokollieren](#)

Jede Protokollgruppe kann nur eine Datenschutzrichtlinie auf Protokollgruppenebene haben, aber diese Richtlinie kann viele verwaltete Datenkennungen angeben, die geprüft und maskiert werden sollen. Das Limit für eine Datenschutzrichtlinie beträgt 30 720 Zeichen.

 Important

Vertrauliche Daten werden erkannt und maskiert, wenn sie in die Protokollgruppe aufgenommen werden. Wenn Sie eine Datenschutzrichtlinie festlegen, werden Protokollereignisse, die vor diesem Zeitpunkt in die Protokollgruppe aufgenommen wurden, nicht maskiert.

CloudWatch Logs unterstützt viele verwaltete Datenkennungen, die vorkonfigurierte Datentypen bieten, die Sie auswählen können, um Finanzdaten, persönliche Gesundheitsinformationen (PHI) und persönlich identifizierbare Informationen (PII) zu schützen. CloudWatch Durch den Datenschutz von Logs können Sie Modelle für Musterabgleich und maschinelles Lernen nutzen, um sensible Daten zu erkennen. Bei einigen Typen verwalteter Datenkennungen hängt die Erkennung davon ab, dass auch bestimmte Schlüsselwörter gefunden werden, die sich in der Nähe der sensiblen Daten befinden. Sie können auch benutzerdefinierte Datenkennungen verwenden, um Datenkennungen zu erstellen, die auf Ihren speziellen Anwendungsfall zugeschnitten sind.

CloudWatch Wenn sensible Daten erkannt werden, wird eine Metrik ausgegeben, die mit den von Ihnen ausgewählten Datenkennungen übereinstimmt. Dies ist die LogEventsWithFindingsMetrik und sie wird im AWS/Logs-Namespace ausgegeben. Sie können diese Metrik verwenden, um CloudWatch Alarmer zu erstellen, und Sie können sie in Diagrammen und Dashboards visualisieren. Die vom Datenschutz ausgegebenen Metriken sind angebotene Metriken und sind kostenlos. Weitere Informationen zu den Metriken, an die CloudWatch Logs sendet CloudWatch, finden Sie unter [Überwachung mit CloudWatch Metriken](#).

Jeder verwaltete Datenbezeichner ist darauf ausgelegt, eine bestimmte Art vertraulicher Daten zu erkennen, z. B. Kreditkartennummern, AWS geheime Zugangsschlüssel oder Passnummern für ein

bestimmtes Land oder eine bestimmte Region. Beim Erstellen einer Datenschutzrichtlinie können Sie es so konfigurieren, dass diese Kennungen von der Protokollgruppe aufgenommene Protokolle analysieren und bei entsprechender Erkennung Maßnahmen ergreifen.

CloudWatch Logs Data Protection kann mithilfe verwalteter Datenkennungen die folgenden Kategorien sensibler Daten erkennen:

- Anmeldeinformationen, wie private Schlüssel oder AWS geheime Zugangsschlüssel
- Finanzinformationen, wie Kreditkartennummern
- Persönlich identifizierbare Informationen (PII), wie Führerscheine oder Sozialversicherungsnummern
- Geschützte Gesundheitsdaten, wie Krankenversicherungs- oder medizinische Identifikationsnummern
- Gerätekennungen, wie IP-Adressen oder MAC-Adressen

Einzelheiten zu den Datentypen, die Sie schützen können, finden Sie unter [Arten von Daten, die Sie schützen können](#).

Inhalt

- [Informationen über Datenschutzrichtlinien](#)
 - [Was sind Datenschutzrichtlinien?](#)
 - [Wie ist die Datenschutzrichtlinie aufgebaut?](#)
 - [JSON-Eigenschaften für die Datenschutzrichtlinie](#)
 - [JSON-Eigenschaften für eine Richtlinienanweisung](#)
 - [JSON-Eigenschaften für eine Richtlinienanweisungsoperation](#)
- [Erforderliche IAM-Berechtigungen zum Erstellen oder Arbeiten mit einer Datenschutzrichtlinie](#)
 - [Für Datenschutzrichtlinien auf Kontoebene sind Berechtigungen erforderlich](#)
 - [Erforderliche Berechtigungen für Datenschutzrichtlinien für eine einzelne Protokollgruppe](#)
 - [Beispiel-Datenschutzrichtlinie](#)
- [Erstellen einer kontoweiten Datenschutzrichtlinie](#)
 - [Konsole](#)
 - [AWS CLI](#)
 - [Syntax der Datenschutzrichtlinie für AWS CLI unsere API-Operationen](#)

- [Erstellen einer Datenschutzrichtlinie für eine einzelne Protokollgruppe](#)
 - [Konsole](#)
 - [AWS CLI](#)
 - [Syntax der Datenschutzrichtlinie für AWS CLI unsere API-Operationen](#)
- [Unmaskierte Daten anzeigen](#)
- [Prüfergebnisberichte](#)
 - [Erforderliche wichtige Richtlinie zum Senden von Prüfungsergebnissen an einen Bucket, der geschützt ist durch AWS KMS](#)
- [Arten von Daten, die Sie schützen können](#)
 - [CloudWatch Protokolliert verwaltete Datenbezeichner für sensible Datentypen](#)
 - [Anmeldeinformationen](#)
 - [Datenbezeichner ARNs für Datentypen mit Anmeldeinformationen](#)
 - [Gerätekennungen](#)
 - [Daten-ID ARNs für Gerätedatentypen](#)
 - [Finanzinformationen](#)
 - [Daten-ID ARNs für Finanzdatentypen](#)
 - [Geschützte Gesundheitsdaten](#)
 - [Datenbezeichner ARNs für Datentypen mit geschützten Gesundheitsinformationen \(PHI\)](#)
 - [Persönlich Identifizierbare Informationen \(PII\)](#)
 - [Schlüsselwörter für Identifikationsnummern des Führerscheins](#)
 - [Schlüsselwörter für nationale Identifikationsnummern](#)
 - [Schlüsselwörter für Passnummern](#)
 - [Schlüsselwörter für Steuerpflichtigenidentifikations- und Referenznummern](#)
 - [Daten-ID ARNs für persönlich identifizierbare Informationen \(PII\)](#)
 - [Benutzerdefinierte Datenbezeichner](#)
 - [Was sind benutzerdefinierte SNS-Datenkennungen?](#)
 - [Einschränkungen für benutzerdefinierte Datenkennungen](#)
 - [Verwenden von benutzerdefinierten Datenbezeichnern in der Konsole](#)

Informationen über Datenschutzrichtlinien

Themen

- [Was sind Datenschutzrichtlinien?](#)
- [Wie ist die Datenschutzrichtlinie aufgebaut?](#)

Was sind Datenschutzrichtlinien?

CloudWatch Logs verwendet Datenschutzrichtlinien, um die vertraulichen Daten auszuwählen, nach denen Sie suchen möchten, und die Maßnahmen, die Sie zum Schutz dieser Daten ergreifen möchten. Um die sensiblen Daten auszuwählen, die für Sie von Interesse sind, verwenden Sie [Datenbezeichner](#). CloudWatch Logs Data Protection erkennt dann die sensiblen Daten mithilfe von maschinellem Lernen und Musterabgleich. Um auf gefundene Daten-IDs zu reagieren, können Sie Vorgänge zum Prüfen und Anonymisieren festlegen. Mit diesen Vorgängen können Sie die gefundenen (oder nicht gefundenen) sensiblen Daten protokollieren und die sensiblen Daten maskieren, wenn die Protokollereignisse angezeigt werden.

Wie ist die Datenschutzrichtlinie aufgebaut?

Wie in der folgenden Abbildung dargestellt, umfasst ein Datenschutzrichtlinien-Dokument die folgenden Elemente:

- Optionale richtlinienweite Informationen oben im Dokument.
- Eine Anweisung, die die Prüfung definiert und Maßnahmen anonymisiert

Pro CloudWatch Logs-Protokollgruppe kann nur eine Datenschutzrichtlinie definiert werden. Die Datenschutzrichtlinie kann mindestens eine Verweigerungs- oder Anonymisierungsanweisung aber nur eine Prüfungsanweisung enthalten.

JSON-Eigenschaften für die Datenschutzrichtlinie

Eine Datenschutzrichtlinie erfordert die folgenden grundlegenden Richtlinieninformationen zur Identifizierung:

- Name – der Name der Richtlinie
- Beschreibung (Optional) – die Beschreibung der Richtlinie
- Version – die Sprachversion der Richtlinie. Die aktuelle Version ist 2021-06-01.

- **Anweisung** – Eine Liste mit Anweisungen, die Aktionen der Datenschutzrichtlinie angeben.

```
{
  "Name": "CloudWatchLogs-PersonalInformation-Protection",
  "Description": "Protect basic types of sensitive data",
  "Version": "2021-06-01",
  "Statement": [
    ...
  ]
}
```

JSON-Eigenschaften für eine Richtlinienanweisung

Eine Richtlinienanweisung legt den Erkennungskontext für den Datenschutzvorgang fest.

- **Sid (Optional)** – der Anweisungsbezeichner
- **DataIdentifier**— Die sensiblen Daten, nach denen CloudWatch Logs suchen soll. Zum Beispiel Name, Adresse oder Telefonnummer.
- **Betrieb** — Die Folgemaßnahmen, entweder „Prüfung“ oder „Anonymisierung“. CloudWatch Logs führt diese Aktionen aus, wenn sensible Daten gefunden werden.

```
{
  ...
  "Statement": [
    {
      "Sid": "audit-policy",
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/Address"
      ],
      "Operation": {
        "Audit": {
          "FindingsDestination": {}
        }
      }
    }
  ],
}
```

JSON-Eigenschaften für eine Richtlinienanweisungsoperation

Eine Richtlinienanweisung legt eine der folgenden Datenschutzoperationen fest.

- Prüfen – Gibt Metriken und Ergebnisberichte aus, ohne die Protokollierung zu unterbrechen. Übereinstimmende Zeichenfolgen erhöhen die LogEventsWithFindingsMetrik, in der CloudWatch Logs im AWS/Logs-Namespace veröffentlicht. CloudWatch Sie können diese Metriken zum Erstellen von Alarmen verwenden.

Ein Beispiel für einen Ergebnisbericht finden Sie unter [Prüfergebnisberichte](#).

Weitere Informationen zu den Metriken, an die CloudWatch Logs sendet, finden Sie unter. CloudWatch [Überwachung mit CloudWatch Metriken](#)

- Anonymisieren – Maskiert die sensiblen Daten, ohne die Protokollierung zu unterbrechen.

Erforderliche IAM-Berechtigungen zum Erstellen oder Arbeiten mit einer Datenschutzrichtlinie

Um mit Datenschutzrichtlinien für Protokollgruppen arbeiten zu können, müssen Sie über bestimmte Berechtigungen verfügen, die in den folgenden Tabellen aufgeführt sind. Die Berechtigungen unterscheiden sich für kontoweite Datenschutzrichtlinien und für Datenschutzrichtlinien, die für eine einzelne Protokollgruppe gelten.

Für Datenschutzrichtlinien auf Kontoebene sind Berechtigungen erforderlich

Note

Wenn Sie einen dieser Vorgänge innerhalb einer Lambda-Funktion ausführen, müssen die Lambda-Ausführungsrolle und die Berechtigungsgrenze auch die folgenden Berechtigungen enthalten.

Operation	IAM-Berechtigung erforderlich	Ressource
Erstellen einer Datenschutzzrichtlinie ohne Audit-Ziele	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*

Operation	IAM-Berechtigung erforderlich	Ressource
Erstellen Sie eine Datenschutzzrichtlinie mit CloudWatch Logs als Auditziel	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*
	logs:CreateLogDelivery	*
	logs:PutResourcePolicy	*
	logs:DescribeResourcePolicies	*
	logs:DescribeLogGroups	*
Erstellen Sie eine Datenschutzzrichtlinie mit Firehose als Auditziel	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*
	logs:CreateLogDelivery	*
	firehose:TagDeliveryStream	arn:aws:logs:::deliverystream/ <i>YOUR_DELIVERY_STREAM</i>
Erstellen einer Datenschutzzrichtlinie mit Amazon S3 als Audit-Ziel	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*

Operation	IAM-Berechtigung erforderlich	Ressource
	<code>logs:CreateLogDelivery</code>	<code>*</code>
	<code>s3:GetBucketPolicy</code>	<code>arn:aws:s3::: <i>YOUR_BUCKET</i></code>
	<code>s3:PutBucketPolicy</code>	<code>arn:aws:s3::: <i>YOUR_BUCKET</i></code>
Aufheben der Maskierung für Protokollereignisse in einer angegebenen Protokollgruppe	<code>logs:Unmask</code>	<code>arn:aws:logs:::log-group:*</code>
Anzeigen einer bestehenden Datenschutzrichtlinie	<code>logs:GetDataProtectionPolicy</code>	<code>*</code>
Löschen einer Datenschutzrichtlinie	<code>logs>DeleteAccountPolicy</code>	<code>*</code>
	<code>logs>DeleteDataProtectionPolicy</code>	<code>*</code>

Wenn bereits Datenschutz-Auditprotokolle an ein Ziel gesendet werden, benötigen andere Richtlinien, die Protokolle an dasselbe Ziel senden, nur die Berechtigungen `logs:PutDataProtectionPolicy` und `logs:CreateLogDelivery`.

Erforderliche Berechtigungen für Datenschutzrichtlinien für eine einzelne Protokollgruppe

Note

Wenn Sie einen dieser Vorgänge innerhalb einer Lambda-Funktion ausführen, müssen die Lambda-Ausführungsrolle und die Berechtigungsgrenze auch die folgenden Berechtigungen enthalten.

Operation	IAM-Berechtigung erforderlich	Ressource
Erstellen einer Datenschutzzrichtlinie ohne Audit-Ziele	logs:PutDataProtectionPolicy	arn:aws:logs::log-group: <i>YOUR_LOG_GROUP</i> :*
Erstellen Sie eine Datenschutzzrichtlinie mit CloudWatch Logs als Auditziel	logs:PutDataProtectionPolicy logs:CreateLogDelivery logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups	arn:aws:logs::log-group: <i>YOUR_LOG_GROUP</i> :* * * * *
Erstellen Sie eine Datenschutzzrichtlinie mit Firehose als Auditziel	logs:PutDataProtectionPolicy logs:CreateLogDelivery firehose:TagDeliveryStream	arn:aws:logs::log-group: <i>YOUR_LOG_GROUP</i> :* * arn:aws:logs::deliverystream/ <i>YOUR_DELIVERY_STREAM</i>
Erstellen einer Datenschutzzrichtlinie mit Amazon S3 als ein Audit-Ziel	logs:PutDataProtectionPolicy logs:CreateLogDelivery s3:GetBucketPolicy	arn:aws:logs::log-group: <i>YOUR_LOG_GROUP</i> :* * arn:aws:s3::: <i>YOUR_BUCKET</i>

Operation	IAM-Berechtigung erforderlich	Ressource
	s3:PutBucketPolicy	arn:aws:s3::: <i>YOUR_BUCKET</i>
Entfernen von Maskierungen bei Protokollereignissen	logs:Unmask	arn:aws:logs:::log-group: <i>YOUR_LOG_GROUP</i> :*
Eine bestehende Datenschutzzrichtlinie ansehen	logs:GetDataProtectionPolicy	arn:aws:logs:::log-group: <i>YOUR_LOG_GROUP</i> :*
Löschen einer Datenschutzzrichtlinie	logs>DeleteDataProtectionPolicy	arn:aws:logs:::log-group: <i>YOUR_LOG_GROUP</i> :*

Wenn bereits Datenschutz-Auditprotokolle an ein Ziel gesendet werden, benötigen andere Richtlinien, die Protokolle an dasselbe Ziel senden, nur die Berechtigungen logs:PutDataProtectionPolicy und logs:CreateLogDelivery.

Beispiel-Datenschutzrichtlinie

Die folgende Beispielrichtlinie ermöglicht es einem Benutzer, Datenschutzrichtlinien zu erstellen, anzuzeigen und zu löschen, die Audit-Ergebnisse an alle drei Arten von Audit-Zielen senden können. Es erlaubt dem Benutzer nicht, unmaskierte Daten einzusehen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowLogDeliveryConfiguration",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:PutResourcePolicy",
```

```

        "logs:DescribeLogGroups",
        "logs:DescribeResourcePolicies"
    ],
    "Resource": "*"
  },
  {
    "Sid": "AllowDataProtectionAndBucketConfiguration",
    "Effect": "Allow",
    "Action": [
      "logs:GetDataProtectionPolicy",
      "logs:DeleteDataProtectionPolicy",
      "logs:PutDataProtectionPolicy",
      "s3:PutBucketPolicy",
      "firehose:TagDeliveryStream",
      "s3:GetBucketPolicy"
    ],
    "Resource": [
      "arn:aws:firehose:us-east-1:111122223333:deliverystream/delivery-stream-name",
      "arn:aws:s3:::amzn-s3-demo-destination-bucket",
      "arn:aws:logs:us-east-1:111122223333:log-group:log-group-name:"
    ]
  }
]
}

```

Erstellen einer kontoweiten Datenschutzrichtlinie

Sie können die CloudWatch Logs-Konsole oder AWS CLI Befehle verwenden, um eine Datenschutzrichtlinie zu erstellen, um sensible Daten für alle Protokollgruppen in Ihrem Konto zu maskieren. Dies wirkt sich sowohl auf aktuelle Protokollgruppen als auch auf Protokollgruppen aus, die Sie in Zukunft erstellen.

Important

Vertrauliche Daten werden erkannt und maskiert, wenn sie in die Protokollgruppe aufgenommen werden. Wenn Sie eine Datenschutzrichtlinie festlegen, werden Protokollereignisse, die vor diesem Zeitpunkt in die Protokollgruppe aufgenommen wurden, nicht maskiert.

Themen

- [Konsole](#)
- [AWS CLI](#)

Konsole

Verwenden der Konsole zum Erstellen einer kontoweiten Datenschutzrichtlinie

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Settings (Einstellungen). Diese Option befindet sich am Ende der Liste.
3. Wählen Sie die Registerkarte Protokolle aus.
4. Wählen Sie Konfigurieren aus.
5. Wählen Sie unter Verwaltete Datenkennungen die Datentypen aus, die Sie für all Ihre Protokollgruppen prüfen und maskieren möchten. Sie können im Auswahlfeld eingeben, welche Kennungen Sie finden möchten.

Wir empfehlen Ihnen, nur die Datenkennungen auszuwählen, die für Ihre Protokolldaten und Ihr Unternehmen relevant sind. Die Auswahl vieler Datentypen kann zu falsch-positiven Ergebnissen führen.

Einzelheiten zu den Datentypen, die Sie schützen können, finden Sie unter [Arten von Daten, die Sie schützen können](#).

6. (Optional) Wenn Sie andere Datentypen mithilfe benutzerdefinierter Datenbezeichner prüfen und maskieren möchten, wählen Sie Benutzerdefinierten Datenbezeichner hinzufügen aus. Geben Sie dann einen Namen für den Datentyp und den regulären Ausdruck ein, mit dem in den Protokollereignissen nach diesem Datentyp gesucht werden soll. Weitere Informationen finden Sie unter [Benutzerdefinierte Datenbezeichner](#).

Eine einzelne Datenschutzrichtlinie kann bis zu 10 benutzerdefinierte Datenkennungen enthalten. Jeder reguläre Ausdruck, der einen benutzerdefinierten Datenbezeichner definiert, muss 200 Zeichen oder weniger lang sein.

7. (Optional) Wählen Sie einen oder mehrere Services aus, an die Sie die Audit-Ergebnisse senden möchten. Auch wenn Sie sich dafür entscheiden, keine Audit-Ergebnisse an einen dieser Services zu senden, werden die von Ihnen ausgewählten vertraulichen Datentypen dennoch maskiert.

8. Wählen Sie **Activate data protection** (Datenschutz aktivieren) aus.

AWS CLI

Um den AWS CLI zu verwenden, um eine Datenschutzrichtlinie zu erstellen

1. Verwenden Sie einen Text-Editor, um eine Richtliniendatei mit dem Namen `DataProtectionPolicy.json` zu erstellen. Informationen zur Richtliniensyntax finden Sie im folgenden Abschnitt.
2. Geben Sie den folgenden Befehl ein:

```
aws logs put-account-policy \
--policy-name TEST_POLICY --policy-type "DATA_PROTECTION_POLICY" \
--policy-document file://policy.json \
--scope "ALL" \
--region us-west-2
```

Syntax der Datenschutzrichtlinie für AWS CLI unsere API-Operationen

Wenn Sie eine JSON-Datenschutzrichtlinie zur Verwendung in einem AWS CLI Befehl oder einer API-Operation erstellen, muss die Richtlinie zwei JSON-Blöcke enthalten:

- Der erste Block muss sowohl ein `DataIdentifier`-Array als auch eine `Operation`-Eigenschaft mit einer `Audit`-Aktion enthalten. Das `DataIdentifier`-Array listet die vertraulichen Datentypen auf, die Sie maskieren möchten. Weitere Informationen zu den verfügbaren Optionen finden Sie unter [Arten von Daten, die Sie schützen können](#).

Die `Operation`-Eigenschaft mit der `Audit`-Aktion ist erforderlich, um die Begriffe für vertrauliche Daten zu finden. Diese `Audit`-Aktion muss ein `FindingsDestination`-Objekt enthalten. Sie können dieses `FindingsDestination`-Objekt optional verwenden, um ein oder mehrere Ziele aufzulisten, an die Prüfergebnisberichte gesendet werden sollen. Wenn Sie Ziele wie Protokollgruppen, Amazon Data Firehose-Streams und S3-Buckets angeben, müssen diese bereits vorhanden sein. Ein Beispiel für einen Prüfergebnisbericht finden Sie unter [Prüfergebnisberichte](#).

- Der zweite Block muss sowohl ein `DataIdentifier`-Array als auch eine `Operation`-Eigenschaft mit einer `Deidentify`-Aktion enthalten. Das `DataIdentifier`-Array muss genau mit dem `DataIdentifier`-Array im ersten Block der Richtlinie übereinstimmen.

Die Operation-Eigenschaft mit der Deidentify-Aktion maskiert die Daten tatsächlich und muss das "MaskConfig": {}-Objekt enthalten. Das "MaskConfig": {}-Objekt muss leer sein.

Im Folgenden finden Sie ein Beispiel für eine Datenschutzrichtlinie, die nur verwaltete Datenkennungen verwendet. Diese Richtlinie maskiert E-Mail-Adressen und Führerscheine in Vereinigte Staaten.

Informationen zu Richtlinien, die benutzerdefinierte Datenkennungen angeben, finden Sie unter [Verwenden benutzerdefinierter Datenkennungen in Ihrer Datenschutzrichtlinie](#).

```
{
  "Name": "data-protection-policy",
  "Description": "test description",
  "Version": "2021-06-01",
  "Statement": [{
    "Sid": "audit-policy",
    "DataIdentifier": [
      "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
      "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
    ],
    "Operation": {
      "Audit": {
        "FindingsDestination": {
          "CloudWatchLogs": {
            "LogGroup": "EXISTING_LOG_GROUP_IN_YOUR_ACCOUNT",
          },
          "Firehose": {
            "DeliveryStream": "EXISTING_STREAM_IN_YOUR_ACCOUNT"
          },
          "S3": {
            "Bucket": "EXISTING_BUCKET"
          }
        }
      }
    }
  },
  {
    "Sid": "redact-policy",
    "DataIdentifier": [
      "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
      "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
    ]
  }
]
```



```
    ],
    "Operation": {
      "Deidentify": {
        "MaskConfig": {}
      }
    }
  }
]
```

Erstellen einer Datenschutzrichtlinie für eine einzelne Protokollgruppe

Sie können die CloudWatch Logs-Konsole oder AWS CLI Befehle verwenden, um eine Datenschutzrichtlinie zum Maskieren vertraulicher Daten zu erstellen.

Sie können jeder Protokollgruppe eine Datenschutzrichtlinie zuweisen. Jede Datenschutzrichtlinie kann mehrere Arten von Informationen überprüfen. Jede Datenschutzrichtlinie kann eine Audit-Anweisung enthalten.

Themen

- [Konsole](#)
- [AWS CLI](#)

Konsole

Verwenden der Konsole zum Erstellen einer Datenschutzrichtlinie

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle), Log groups (Protokollgruppen) aus.
3. Wählen Sie den Namen der Protokollgruppe aus.
4. Wählen Sie Actions (Aktionen), Create data protection policy (Datenschutzrichtlinie erstellen) aus.
5. Wählen Sie unter Verwaltete Datenbezeichner die Datentypen aus, die Sie in dieser Protokollgruppe prüfen und maskieren möchten. Sie können im Auswahlfeld eingeben, welche Kennungen Sie finden möchten.

Wir empfehlen Ihnen, nur die Datenkennungen auszuwählen, die für Ihre Protokolldaten und Ihr Unternehmen relevant sind. Die Auswahl vieler Datentypen kann zu falsch-positiven Ergebnissen führen.

Einzelheiten dazu, welche Datentypen Sie mithilfe verwalteter Datenbezeichner schützen können, finden Sie unter [Arten von Daten, die Sie schützen können](#)

6. (Optional) Wenn Sie andere Datentypen mithilfe benutzerdefinierter Datenbezeichner prüfen und maskieren möchten, wählen Sie Benutzerdefinierten Datenbezeichner hinzufügen. Geben Sie dann einen Namen für den Datentyp und den regulären Ausdruck ein, mit dem in den Protokollereignissen nach diesem Datentyp gesucht werden soll. Weitere Informationen finden Sie unter [Benutzerdefinierte Datenbezeichner](#).

Eine einzelne Datenschutzrichtlinie kann bis zu 10 benutzerdefinierte Datenkennungen enthalten. Jeder reguläre Ausdruck, der einen benutzerdefinierten Datenbezeichner definiert, muss 200 Zeichen oder weniger lang sein.

7. (Optional) Wählen Sie einen oder mehrere Services aus, an die Sie die Audit-Ergebnisse senden möchten. Auch wenn Sie sich dafür entscheiden, keine Audit-Ergebnisse an einen dieser Services zu senden, werden die von Ihnen ausgewählten vertraulichen Datentypen dennoch maskiert.
8. Wählen Sie Activate data protection (Datenschutz aktivieren) aus.

AWS CLI

Um den AWS CLI zu verwenden, um eine Datenschutzrichtlinie zu erstellen

1. Verwenden Sie einen Text-Editor, um eine Richtliniendatei mit dem Namen `DataProtectionPolicy.json` zu erstellen. Informationen zur Richtliniensyntax finden Sie im folgenden Abschnitt.
2. Geben Sie den folgenden Befehl ein:

```
aws logs put-data-protection-policy --log-group-identifier "my-log-group" --policy-document file:///Path/DataProtectionPolicy.json --region us-west-2
```

Syntax der Datenschutzrichtlinie für AWS CLI unsere API-Operationen

Wenn Sie eine JSON-Datenschutzrichtlinie zur Verwendung in einem AWS CLI Befehl oder einer API-Operation erstellen, muss die Richtlinie zwei JSON-Blöcke enthalten:

- Der erste Block muss sowohl ein `DataIdentifier`-Array als auch eine `Operation`-Eigenschaft mit einer `Audit`-Aktion enthalten. Das `DataIdentifier`-Array listet die vertraulichen Datentypen auf, die Sie maskieren möchten. Weitere Informationen zu den verfügbaren Optionen finden Sie unter [Arten von Daten, die Sie schützen können](#).

Die `Operation`-Eigenschaft mit der `Audit`-Aktion ist erforderlich, um die Begriffe für vertrauliche Daten zu finden. Diese `Audit`-Aktion muss ein `FindingsDestination`-Objekt enthalten. Sie können dieses `FindingsDestination`-Objekt optional verwenden, um ein oder mehrere Ziele aufzulisten, an die Prüfergebnisberichte gesendet werden sollen. Wenn Sie Ziele wie Protokollgruppen, Amazon Data Firehose-Streams und S3-Buckets angeben, müssen diese bereits vorhanden sein. Ein Beispiel für einen Prüfergebnisbericht finden Sie unter [Prüfergebnisberichte](#).

- Der zweite Block muss sowohl ein `DataIdentifier`-Array als auch eine `Operation`-Eigenschaft mit einer `Deidentify`-Aktion enthalten. Das `DataIdentifier`-Array muss genau mit dem `DataIdentifier`-Array im ersten Block der Richtlinie übereinstimmen.

Die `Operation`-Eigenschaft mit der `Deidentify`-Aktion maskiert die Daten tatsächlich und muss das `"MaskConfig": {}`-Objekt enthalten. Das `"MaskConfig": {}`-Objekt muss leer sein.

Im Folgenden finden Sie ein Beispiel für eine Datenschutzrichtlinie, die E-Mail-Adressen und Führerscheine der Vereinigten Staaten maskiert.

```
{
  "Name": "data-protection-policy",
  "Description": "test description",
  "Version": "2021-06-01",
  "Statement": [{
    "Sid": "audit-policy",
    "DataIdentifier": [
      "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
      "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
    ],
    "Operation": {
      "Audit": {
        "FindingsDestination": {
          "CloudWatchLogs": {
```

```

        "LogGroup": "EXISTING_LOG_GROUP_IN_YOUR_ACCOUNT",
    },
    "Firehose": {
        "DeliveryStream": "EXISTING_STREAM_IN_YOUR_ACCOUNT"
    },
    "S3": {
        "Bucket": "EXISTING_BUCKET"
    }
}
}
}
},
{
    "Sid": "redact-policy",
    "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
        "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
    ],
    "Operation": {
        "Deidentify": {
            "MaskConfig": {}
        }
    }
}
}
]
}

```

Unmaskierte Daten anzeigen

Um unmaskierte Daten anzeigen zu können, muss ein Benutzer über die `logs:Unmask`-Berechtigung verfügen. Benutzer mit dieser Berechtigung können die unmaskierten Daten auf folgende Weise sehen:

- Wenn Sie sich die Ereignisse in einem Protokollstream ansehen, wählen Sie **Display (Anzeigen)**, **Unmask (Maskierung aufheben)** aus.
- Verwenden Sie eine CloudWatch Logs Insights-Abfrage, die den Befehl `unmask (@message)` enthält. In der folgenden Beispielabfrage werden die 20 letzten Protokollereignisse im Stream unmaskiert angezeigt:

```

fields @timestamp, @message, unmask(@message)
| sort @timestamp desc

```

```
| limit 20
```

Weitere Informationen zu CloudWatch Logs Insights-Befehlen finden Sie unter [CloudWatch Abfragesyntax in Logs Insights](#).

- Verwenden Sie eine [GetLogEvents](#) [FilterLogEvents](#) Oder-Operation mit dem unmask Parameter.

Die CloudWatchLogsFullAccessRichtlinie beinhaltet die logs:Unmask Genehmigung. Um einem Benutzer logs:Unmask zu gewähren, der dies nicht hat CloudWatchLogsFullAccess, können Sie diesem Benutzer eine benutzerdefinierte IAM-Richtlinie anhängen. Weitere Informationen finden Sie unter [Hinzufügen von Berechtigungen zu einem Benutzer \(Konsole\)](#).

Prüfergebnisberichte

Wenn Sie die Datenschutzprüfungsrichtlinien für CloudWatch Logs einrichten, um Auditberichte in CloudWatch Logs, Amazon S3 oder Firehose zu schreiben, ähneln diese Ergebnisberichte dem folgenden Beispiel. CloudWatch Logs schreibt für jedes Protokollereignis, das vertrauliche Daten enthält, einen Ergebnisbericht.

```
{
  "auditTimestamp": "2023-01-23T21:11:20Z",
  "resourceArn": "arn:aws:logs:us-west-2:111122223333:log-group:/aws/lambda/MyLogGroup:*",
  "dataIdentifiers": [
    {
      "name": "EmailAddress",
      "count": 2,
      "detections": [
        {
          "start": 13,
          "end": 26
        },
        {
          "start": 30,
          "end": 43
        }
      ]
    }
  ]
}
```

Die Felder im Bericht lauten wie folgt:

- Das Feld `resourceArn` zeigt die Protokollgruppe an, in der die sensiblen Daten gefunden wurden.
- Das Objekt `dataIdentifiers` zeigt Informationen zu den Ergebnissen für einen sensiblen Datentyp an, den Sie gerade prüfen.
- Das Feld `name` gibt an, über welche Art von sensiblen Daten dieser Abschnitt berichtet.
- Das Feld `count` zeigt an, wie oft diese Art von sensiblen Daten im Protokollereignis vorkommt.
- Die Felder `start` und `end` zeigen, wo im Protokollereignis, nach Zeichenanzahl, jedes Vorkommen der sensiblen Daten erscheint.

Das vorherige Beispiel zeigt einen Bericht über das Auffinden von zwei E-Mail-Adressen in einem Protokollereignis. Die erste E-Mail-Adresse beginnt mit dem 13. Zeichen des Protokollereignisses und endet mit dem 26. Zeichen. Die zweite E-Mail-Adresse reicht vom 30. bis zum 43. Zeichen. Obwohl dieses Protokollereignis zwei E-Mail-Adressen hat, wird der Wert der `LogEventsWithFindings`-Metrik nur um eins erhöht, da diese Metrik die Anzahl der Protokollereignisse zählt, die vertrauliche Daten enthalten, nicht die Anzahl der Vorkommen vertraulicher Daten.

Erforderliche wichtige Richtlinie zum Senden von Prüfungsergebnissen an einen Bucket, der geschützt ist durch AWS KMS

Sie können die Daten in einem Amazon-S3-Bucket schützen, indem Sie entweder die serverseitige Verschlüsselung mit von Amazon S3 verwalteten Schlüsseln (SSE-S3) oder die serverseitige Verschlüsselung mit KMS-Schlüsseln (SSE-KMS) aktivieren. Weitere Informationen finden Sie unter [Schützen von Daten mithilfe serverseitiger Verschlüsselung](#) im Amazon-S3-Entwicklerhandbuch.

Wenn Sie Audit-Ergebnisse an einen Bucket mit SSE-S3-Schutz senden, ist keine zusätzliche Konfiguration erforderlich. Amazon S3 verarbeitet den Verschlüsselungsschlüssel.

Wenn Sie Prüfungsergebnisse an einen Bucket mit SSE-KMS-Schutz senden, müssen Sie die Schlüsselrichtlinie für Ihren KMS-Schlüssel aktualisieren, damit das Protokollzustellungskonto in Ihren S3-Bucket schreiben kann. Weitere Informationen zur erforderlichen Schlüsselrichtlinie für die Verwendung mit SSE-KMS finden Sie [Amazon S3](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

Arten von Daten, die Sie schützen können

Dieser Abschnitt enthält Informationen zu den Datentypen, die Sie mit einer CloudWatch Logs-Datenschutzrichtlinie schützen können. CloudWatch Logs Managed Data Identifier bieten

vorkonfigurierte Datentypen zum Schutz von Finanzdaten, persönlichen Gesundheitsinformationen (PHI) und persönlich identifizierbaren Informationen (PII). Sie können auch benutzerdefinierte Datenkennungen verwenden, um Datenkennungen zu erstellen, die auf Ihren speziellen Anwendungsfall zugeschnitten sind.

Inhalt

- [CloudWatch Protokolliert verwaltete Datenbezeichner für sensible Datentypen](#)
 - [Anmeldeinformationen](#)
 - [Datenbezeichner ARNs für Datentypen mit Anmeldeinformationen](#)
 - [Gerätekennungen](#)
 - [Daten-ID ARNs für Gerätedatentypen](#)
 - [Finanzinformationen](#)
 - [Daten-ID ARNs für Finanzdatentypen](#)
 - [Geschützte Gesundheitsdaten](#)
 - [Datenbezeichner ARNs für Datentypen mit geschützten Gesundheitsinformationen \(PHI\)](#)
 - [Persönlich Identifizierbare Informationen \(PII\)](#)
 - [Schlüsselwörter für Identifikationsnummern des Führerscheins](#)
 - [Schlüsselwörter für nationale Identifikationsnummern](#)
 - [Schlüsselwörter für Passnummern](#)
 - [Schlüsselwörter für Steuerpflichtigenidentifikations- und Referenznummern](#)
 - [Daten-ID ARNs für persönlich identifizierbare Informationen \(PII\)](#)
- [Benutzerdefinierte Datenbezeichner](#)
 - [Was sind benutzerdefinierte SNS-Datenkennungen?](#)
 - [Einschränkungen für benutzerdefinierte Datenkennungen](#)
 - [Verwenden von benutzerdefinierten Datenbezeichnern in der Konsole](#)
 - [Verwenden benutzerdefinierter Datenkennungen in Ihrer Datenschutzerklärung](#)

CloudWatch Protokolliert verwaltete Datenbezeichner für sensible Datentypen

Dieser Abschnitt enthält Informationen zu den Datentypen, die Sie mithilfe verwalteter Datenkennungen schützen können, und darüber, welche Länder und Regionen für die einzelnen Datentypen relevant sind.

Arten von Daten, die Sie schützen können

Bei einigen Arten vertraulicher Daten sucht CloudWatch Logs Data Protection nach Schlüsselwörtern in der Nähe der Daten und findet nur dann eine Übereinstimmung, wenn dieses Schlüsselwort gefunden wird. Wenn ein Schlüsselwort in der Nähe eines bestimmten Datentyps stehen muss, muss das Schlüsselwort in der Regel nicht weiter als 30 Zeichen (einschließlich) von den Daten entfernt sein.

Wenn ein Schlüsselwort ein Leerzeichen enthält, CloudWatch sucht Logs Data Protection automatisch nach Schlüsselwortvarianten, bei denen das Leerzeichen fehlt oder die anstelle des Leerzeichens einen Unterstrich (_) oder Bindestrich (-) enthalten. In einigen Fällen erweitert oder kürzt CloudWatch Logs ein Schlüsselwort auch ab, um häufig verwendete Varianten des Schlüsselworts zu berücksichtigen.

In den folgenden Tabellen sind die Typen von Anmelde-, Geräte-, Finanz-, medizinischen und geschützten Gesundheitsinformationen (PHI) aufgeführt, die CloudWatch Logs anhand verwalteter Datenkennungen erkennen kann. Diese gelten zusätzlich zu bestimmten Arten von Daten, die auch als persönlich identifizierbare Informationen (PII) gelten können.

Unterstützte Kennungen, die sprach- und regionsunabhängig sind

Kennung	Kategorie
Address	Persönlich
AwsSecretKey	Anmeldeinformationen
CreditCardExpiration	Finanzanwendungen
CreditCardNumber	Finanzanwendungen
CreditCardSecurityCode	Finanzanwendungen
EmailAddress	Persönlich
IpAddress	Persönlich
LatLong	Persönlich
Name	Persönlich
OpenSshPrivateKey	Anmeldeinformationen

Kennung	Kategorie
PgpPrivateKey	Anmeldeinformationen
PkcsPrivateKey	Anmeldeinformationen
PuttyPrivateKey	Anmeldeinformationen
VehicleIdentificationNumber	Persönlich

Für regionsabhängige Datenkennungen sind der Kennungsname mit einem Bindestrich und die aus zwei Buchstaben bestehenden Codes (ISO 3166-1 Alpha-2) erforderlich. Beispiel, DriversLicense-US.

Unterstützte Kennungen, die einen aus zwei Buchstaben bestehenden Landes- oder Regionalcode enthalten müssen

Kennung	Kategorie	Länder und Sprachen
BankAccountNumber	Finanzanwendungen	DE, ES, FR, GB, IT, US
CepCode	Persönlich	BR
Cnpj	Persönlich	BR
CpfCode	Persönlich	BR
DriversLicense	Persönlich	AT, AU, BE, BG, CA, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IT, LT, LU, LV, MT, NL, PL, PT, RO, SE, SI, SK, US
DrugEnforcementAgencyNumber	Gesundheit	US
ElectoralRollNumber	Persönlich	GB
HealthInsuranceCardNumber	Gesundheit	EU

Kennung	Kategorie	Länder und Sprachen
HealthInsuranceClaimNumber	Gesundheit	US
HealthInsuranceNumber	Gesundheit	FR
HealthcareProcedureCode	Gesundheit	US
IndividualTaxIdentification Number	Persönlich	US
InseeCode	Persönlich	FR
MedicareBeneficiaryNumber	Gesundheit	US
NationalDrugCode	Gesundheit	US
NationalIdentificationNumber	Persönlich	DE, ES, IT
NationalInsuranceNumber	Persönlich	GB
NationalProviderId	Gesundheit	US
NhsNumber	Gesundheit	GB
NieNumber	Persönlich	ES
NifNumber	Persönlich	ES
PassportNumber	Persönlich	CA, DE, ES, FR, GB, IT, US
PermanentResidenceNumber	Persönlich	CA
PersonalHealthNumber	Gesundheit	CA
PhoneNumber	Persönlich	BR, DE, ES, FR, GB, IT, US
PostalCode	Persönlich	CA
RgNumber	Persönlich	BR
SocialInsuranceNumber	Persönlich	CA

Kennung	Kategorie	Länder und Sprachen
Ssn	Persönlich	ES, US
TaxId	Persönlich	DE, ES, FR, GB
ZipCode	Persönlich	US

Anmeldeinformationen

CloudWatch Logs Data Protection kann die folgenden Arten von Anmeldeinformationen finden.

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen
AWS geheimer Zugriffsschlüssel	AwsSecretKey	aws_secret_access_key , credentials , secret access key, secret key, set-awscredential	Alle
Privater OpenSSH-Schlüssel	OpenSSHPrivateKey	Keine	Alle
Privater PGP-Schlüssel	PgpPrivateKey	Keine	Alle
Privater Pkcs-Schlüssel	PkcsPrivateKey	Keine	Alle
Privater PuTTY-Schlüssel	PuttyPrivateKey	Keine	Alle

Datenbezeichner ARNs für Datentypen mit Anmeldeinformationen

Im Folgenden sind die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgeführt, die Sie zu Ihren Datenschutzrichtlinien hinzufügen können.

Daten-ID für Anmeldeinformationen ARNs

```
arn:aws:dataprotection::aws:data-identifier/AwsSecretKey
```

```
arn:aws:dataprotection::aws:data-identifier/OpenSshPrivateKey
```

```
arn:aws:dataprotection::aws:data-identifier/PgpPrivateKey
```

```
arn:aws:dataprotection::aws:data-identifier/PkcsPrivateKey
```

```
arn:aws:dataprotection::aws:data-identifier/PuttyPrivateKey
```

Gerätekennungen

CloudWatch Logs Data Protection kann die folgenden Arten von Gerätekennungen finden.

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen
IP-Adresse	IpAddress	Keine	Alle

Daten-ID ARNs für Gerätedatentypen

Im Folgenden sind die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgeführt, die Sie zu Ihren Datenschutzrichtlinien hinzufügen können.

Gerätedatenkennungs-ARN

```
arn:aws:dataprotection::aws:data-identifier/IpAddress
```

Finanzinformationen

CloudWatch Im Bereich Logs Data Protection finden Sie die folgenden Arten von Finanzinformationen.

Wenn Sie eine Datenschutzrichtlinie festlegen, sucht CloudWatch Logs unabhängig davon, an welchem Standort sich die Protokollgruppe befindet, nach den von Ihnen angegebenen Datenkennungen. Die Informationen in der Spalte Countries and regions (Länder und Regionen) in dieser Tabelle geben an, ob zweibuchstabige Ländercodes an die Datenkennung angehängt werden müssen, um die entsprechenden Schlüsselwörter für diese Länder und Regionen zu ermitteln.

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Bankkontonummer	BankAccountNumber	Ja. Für verschiedene Länder gelten verschiedene Schlüsselwörter. Einzelheiten finden Sie in der Tabelle Keywords for bank account numbers (Schlüsselwörter für Bankkontonummern) weiter unten in diesem Abschnitt.	Deutschland, Frankreich, Vereinigtes Königreich, Italien, Spanien, Vereinigte Staaten	Beinhalten internationale Bankkontonummern (IBANs), die aus bis zu 34 alphanumerischen Zeichen bestehen, einschließlich Elementen wie Ländercodes.
Ablaufdatum der Kreditkarte	CreditCardExpiration	exp d, exp m, exp y, expiration , expiry	Alle	
Kreditkartennummer	CreditCardNumber	account number, american express,	Alle	Für die Erkennung

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
		amex, bank card, card, card number, card num, cc #, ccn, check card, credit, credit card#, dankort, debit, debit card, diners club, discover, electron, japanese card bureau, jcb, mastercard , mc, pan, payment account number, payment card number, pcn, union pay, visa		müssen die Daten eine 13—19-stellige Sequenz sein, die der Luhn-Scheckformel entspricht, und für jede der folgenden Arten von Kreditkarten ein Standardpräfix für Kartennummern verwenden: American Express, Dankort,

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
				Diner's Club, Discover, Electron, Japanese Card Bureau (JCB), Mastercard und Visa. UnionPay
Bestätigungscode für die Kreditkarte	CreditCardSecurity Code	card id, card identification code, card identification number , card security code, card validation code , card validation number , card verification data , card verification value, cvc, cvc2, cvv, cvv2, elo verification code	Alle	

Schlüsselwörter für Bankkontonummern

Verwenden Sie die folgenden Schlüsselwörter für Bankkontonummern. Dazu gehören internationale Bankkontonummern (IBANs), die aus bis zu 34 alphanumerischen Zeichen bestehen, einschließlich Elementen wie Ländercodes.

Land	Schlüsselwörter
Frankreich	account code, account number, accountno# , accountnumber# , bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank account id, iban, numéro de compte
Deutschland	account code, account number, accountno# , accountnumber# , bankleitzahl , bban, customer account id, customer account number, customer bank account id, geheimzahl , iban, kartenummer , kontonummer , kreditkartennummer , sepa
Italien	account code, account number, accountno# , accountnumber# , bban, codice bancario, conto bancario, customer account id, customer account number, customer bank account id, iban, numero di conto
Spanien	account code, account number, accountno# , accountnumber# , bban, código cuenta, código cuenta bancaria, cuenta cliente id, customer account ID, customer account number, customer bank account id, iban, número cuenta bancaria cliente, número cuenta cliente
Großbritannien und Nordirland	account code, account number, accountno# , accountnumber# , bban, customer account ID, customer account number, customer bank account id, iban, sepa
Vereinigte Staaten	bank account, bank acct, checking account, checking acct, deposit account, deposit acct, savings account, savings acct, chequing account, chequing acct

CloudWatch Logs meldet keine Vorkommen der folgenden Sequenzen, die Kreditkartenaussteller öffentlichen Tests vorbehalten haben.

```
1220000000000003, 2222405343248877, 2222990905257051, 2223007648726984,
2223577120017656,
30569309025904, 34343434343434, 3528000700000000, 3530111333300000, 3566002020360505,
36148900647913,
```



```
36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237,
4012888888881881,
4111111111111111, 4222222222222, 4444333322221111, 4462030000000000, 4484070000000000,
49118300000000,
4917300800000000, 4917610000000000, 4917610000000000003, 5019717010103742,
5105105105105100,
5111010030175156, 5185540810000019, 5200828282828210, 5204230080000017,
5204740009900014, 5420923878724339,
5454545454545454, 5455330760000018, 55069004900000436, 55069004900000444,
55069005100000234, 5506920809243667,
5506922400634930, 5506927427317625, 5553042241984105, 5555553753048194,
5555555555554444, 5610591081018250,
6011000990139424, 6011000400000000, 6011111111111117, 630490017740292441,
630495060000000000,
6331101999990016, 6759649826438453, 6799990100000000019, and 76009244561.
```

Daten-ID ARNs für Finanzdatentypen

Im Folgenden sind die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgeführt, die Sie zu Ihren Datenschutzrichtlinien hinzufügen können.

Kennung für Finanzdaten ARNs

```
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-DE
```

```
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-ES
```

```
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-FR
```

```
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-GB
```

```
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-IT
```

```
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-US
```

```
arn:aws:dataprotection::aws:data-identifier/CreditCardExpiration
```

```
arn:aws:dataprotection::aws:data-identifier/CreditCardNumber
```

```
arn:aws:dataprotection::aws:data-identifier/CreditCardSecurityCode
```

Geschützte Gesundheitsdaten

CloudWatch Der Datenschutz protokolliert die folgenden Arten geschützter Gesundheitsinformationen (PHI).

Wenn Sie eine Datenschutzrichtlinie festlegen, sucht CloudWatch Logs unabhängig davon, an welchem Standort sich die Protokollgruppe befindet, nach den von Ihnen angegebenen Datenbezeichnern. Die Informationen in der Spalte Countries and regions (Länder und Regionen) in dieser Tabelle geben an, ob zweibuchstabige Ländercodes an die Datenkennung angehängt werden müssen, um die entsprechenden Schlüsselwörter für diese Länder und Regionen zu ermitteln.

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen
Registrierungsnummer der Drug Enforcement Agency (DEA)	DrugEnforcementAgencyNumber	dea number, dea registration	Vereinigte Staaten
Nummer der Krankenversicherungskarte (EHIC)	HealthInsuranceCardNumber	assicurazione sanitaria numero, carta assicurazione numero, carte d'assurance maladie , carte européenne d'assurance maladie , ceam, ehic, ehic#, finlandehicnumber# , gesundheitskarte , hälsokort , health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte , krankenversicherungsnummer , medical account number,	Europäische Union

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen
		numero conto medico, numéro d'assurance maladie, numéro de carte d'assurance, numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanho itokortin, sairausva kuutuskortti, sairausvakuutusnumero, sjukförsäkringsnummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveystkortti, tessera sanitaria assicurazione numero, versicherungsnummer	
Health Insurance Claim Number (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hichn, hichn#, hichno#	Vereinigte Staaten
Krankenversicherungs- oder medizinische Identifizierungsnummer	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	Frankreich

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen
Standardisierte Codes für medizinische Leistungen (HCPCS)	HealthcareProcedureCode	current procedural terminology , hcpcs, healthcare common procedure coding system	Vereinigte Staaten
Medicare-Versichertennummer (MBN)	MedicareBeneficiaryNumber	mbi, medicare beneficiary	Vereinigte Staaten
National Drug Code (NDC)	NationalDrugCode	national drug code, ndc	Vereinigte Staaten
National Provider Identifier (NPI)	NationalProviderId	hipaa, n.p.i., national provider, npi	Vereinigte Staaten
Nummer des National Health Service (NHS)	NhsNumber	national health service, NHS	Großbritannien
Persönliche Gesundheitsnummer	PersonalHealthNumber	canada healthcare number, msp number, care number, phn, soins de santé	Kanada

Datenbezeichner ARNs für Datentypen mit geschützten Gesundheitsinformationen (PHI)

Im Folgenden ist die Daten-ID Amazon Resource Names (ARNs) aufgeführt, die in Datenschutzrichtlinien für geschützte Gesundheitsinformationen (PHI) verwendet werden kann.

PHI-Datenbezeichner ARNs

```
arn:aws:dataprotection::aws:data-identifier/DrugEnforcementAgencyNumber-US
```

```
arn:aws:dataprotection::aws:data-identifier/HealthcareProcedureCode-US
```

```
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceCardNumber-EU
```

```
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceClaimNumber-US
```

```
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceNumber-FR
```

```
arn:aws:dataprotection::aws:data-identifier/MedicareBeneficiaryNumber-US
```

```
arn:aws:dataprotection::aws:data-identifier/NationalDrugCode-US
```

```
arn:aws:dataprotection::aws:data-identifier/NationalInsuranceNumber-GB
```

```
arn:aws:dataprotection::aws:data-identifier/NationalProviderId-US
```

```
arn:aws:dataprotection::aws:data-identifier/NhsNumber-GB
```

```
arn:aws:dataprotection::aws:data-identifier/PersonalHealthNumber-CA
```

Persönlich Identifizierbare Informationen (PII)

CloudWatch Der Datenschutz protokolliert die folgenden Arten von personenbezogenen Daten (PII).

Wenn Sie eine Datenschutzrichtlinie festlegen, sucht CloudWatch Logs unabhängig davon, an welchem Standort sich die Protokollgruppe befindet, nach den von Ihnen angegebenen Datenkennungen. Die Informationen in der Spalte Countries and regions (Länder und Regionen) in

dieser Tabelle geben an, ob zweibuchstabile Ländercodes an die Datenkennung angehängt werden müssen, um die entsprechenden Schlüsselwörter für diese Länder und Regionen zu ermitteln.

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Geburtsdatum	DateOfBirth	dob, date of birth, birthdate , birth date, birthday, b-day, bday	Beliebig	Der Support umfasst die meisten Datumsformate, z. B. nur Ziffern und Kombinationen aus Ziffern und Monatsnamen. Datumskomponenten können durch Leerzeichen, Schrägstriche (/) oder Bindestriche (-)

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
				getrennt werden.
Código de Endereçamento Postal (CEP)	CepCode	cep, código de endereçamento postal, código de endereçamento postal	Brasilien	
Cadastro Nacional da Pessoa Jurídica (CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa jurídica, cnpj	Brasilien	
Cadastro de Pessoas Físicas (CPF)	CpfCode	Cadastro de pessoas físicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa física, cpf	Brasilien	

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Identifikationsnummer des Führerscheins	DriversLicense	Ja. Für verschiedene Länder gelten verschiedene Schlüsselwörter. Einzelheiten finden Sie in der Tabelle Drivers license identification numbers (Führerschein-Identifikationsnummern) weiter unten in diesem Abschnitt.	Viele Länder. Einzelheiten finden Sie in der Tabelle Drivers license identification numbers (Führerschein-Identifikationsnummern).	
Nummer der Wählerliste	Electoral RollNumber	electoral #, electoral number, electoral roll #, electoral roll no., electoral roll number, electoral rollno	Großbritannien und Nordirland	

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Individuelle Steuerpflichtigen-ID	IndividualTaxIdentificationNumber	Ja. Für verschiedene Länder gelten verschiedene Schlüsselwörter. Einzelheiten finden Sie in der Tabelle Individual taxpayer identification numbers (Individuelle Steuerzahler-Identifikationsnummern) weiter unten in diesem Abschnitt.	Brasilien, Frankreich, Deutschland, Spanien, Großbritannien	
Nationales Institut für Statistik und Wirtschaftsstudien (INSEE)	InseeCode	Ja. Für verschiedene Länder gelten verschiedene Schlüsselwörter. Einzelheiten finden Sie in der Tabelle Keywords for national identification numbers (Schlüsselwörter für nationale Identifikationsnummern) weiter unten in diesem Abschnitt.	Frankreich	

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Nationale Identifikationsnummern	NationalIdentificationNumber	Ja. Einzelheiten finden Sie in der Tabelle Keywords for national identification numbers (Schlüsselwörter für nationale Identifikationsnummern) weiter unten in diesem Abschnitt.	Deutschland, Italien, Spanien	Dazu gehören Kennungen des Documento Nacional de Identidad (DNI) (Spanien), , Codice fiscale codes (Italien) und Personalausweisnummern (Deutsch).
Landesversicherungsnummer (NINO)	NationalInsuranceNumber	insurance no., insurance number, insurance# , national insurance number, nationalinsurance# , nationalinsurance number , nin, nino	Großbritannien und Nordirland	–

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Número de identidad de extranjero (NIE)	NieNumber	Ja. Für verschiedene Länder gelten verschiedene Schlüsselwörter. Einzelheiten finden Sie in der Tabelle Individual taxpayer identification numbers (Individuelle Steuerzahler-Identifikationsnummern) weiter unten in diesem Abschnitt.	Spanien	
Número de Identificación Fiscal (NIF)	NifNumber	Ja. Für verschiedene Länder gelten verschiedene Schlüsselwörter. Einzelheiten finden Sie in der Tabelle Individual taxpayer identification numbers (Individuelle Steuerzahler-Identifikationsnummern) weiter unten in diesem Abschnitt.	Spanien	
Passnummer	PassportNumber	Ja. Für verschiedene Länder gelten verschiedene Schlüsselwörter. Einzelheiten finden Sie in der Tabelle Keywords for passport numbers (Schlüsselwörter für Reisepassnummern) weiter unten in diesem Abschnitt.	Kanada, Frankreich, Deutschland, Italien, Spanien, Großbritannien, USA	

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Ständige Wohnsitznummer	Permanent Residence Number	carte résident permanent , numéro carte résident permanent , numéro résident permanent , permanent resident card, permanent resident card number, permanent resident no, permanent resident no., permanent resident number, pr no, pr no., pr non, pr number, résident permanent no., résident permanent non	Kanada	

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Phone number (Telefonnummer)	PhoneNumber	Brasilien: Zu den Schlüsselwörtern gehören auch: cel, celular, fone, móvel, número residencial, numero residencial, telefone Andere: cell, contact, fax, fax number, mobile, phone, phone number, tel, telephone, telephone number	Brasilien, Kanada, Frankreich, Deutschland, Italien, Spanien, Großbritannien, USA	Dazu gehören gebührenfreie Nummern in den USA und Faxnummern. Wenn sich ein Schlüsselwort in der Nähe der Daten befindet, muss die Nummer keine Landeswahl enthalten. Wenn sich ein Schlüsselwort

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
				nicht in der Nähe der Daten befindet, muss die Nummer eine Landeswahl enthalten.
Postal Code (Postleitzahl)	PostalCode	Keine	Kanada	
Registro Geral (RG)	RgNumber	Ja. Für verschiedene Länder gelten verschiedene Schlüsselwörter. Einzelheiten finden Sie in der Tabelle Individual taxpayer identification numbers (Individuelle Steuerzahler-Identifikationsnummern) weiter unten in diesem Abschnitt.	Brasilien	
Sozialversicherungsnummer (SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	Kanada	

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Sozialversicherungsnummer (SSN)	Ssn	Spanien – número de la seguridad social, social security no., social security no, número de la seguridad social, social security number, social securityno# , ssn, ssn# USA – social security, ss#, ssn	Spanien, USA	
Steuerpflichtigen-Identifikationsnummer oder Referenznummer	TaxId	Ja. Für verschiedene Länder gelten verschiedene Schlüsselwörter. Einzelheiten finden Sie in der Tabelle Individual taxpayer identification numbers (Individuelle Steuerzahler-Identifikationsnummern) weiter unten in diesem Abschnitt.	Frankreich, Deutschland, Spanien, Großbritannien	Dazu gehören TIN (Frankreich), Steueridentifikationsnummer (Deutschland), CIF (Spanien) und TRN, UTR (Großbritannien).

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Zip code (Postleitzahl)	ZipCode	zip code, zip+4	Vereinigte Staaten	Postleitzahl der USA.

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Postanschrift	Address	Keine	Australien, Kanada, Frankreich, Deutschland, Italien, Spanien, Großbritannien, USA	Obwohl ein Schlüsselwort nicht erforderlich ist, muss die Adresse bei der Erkennung den Namen einer Stadt oder eines Ortes sowie einen Zip-Code oder eine Postleitzahl enthalten.
E-Mail-Adresse	EmailAddress	Keine	Beliebig	

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Koordinaten des globalen Positionierungssystems (GPS)	LatLong	coordinate , coordinates , lat long, latitude longitude , location, position	Beliebig	CloudWatch Logs können GPS-Koordinaten erkennen, wenn die Breiten- und Längengradkoordinaten paarweise gespeichert werden und sie im Format Decimal Degrees (DD) vorliegen, z. B. 41.948614 , -87.655311. Nicht unterstützt

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
				zt werden Koordinaten im DDM-Format (Grad, Dezimalstelle, Minuten), z. B. 41°56.9168'N 87°39.3187'W, oder im DMS-Format (Grad, Minuten, Sekunden) , z. B. 41°56'55.0104"N 87°39'19.1196"W.

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Vollständiger Name	Name	Keine	Beliebig	CloudWatch In Protokollen können nur vollständige Namen erkannt werden. Unterstützt werden nur lateinische Zeichensätze.

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen	Hinweise
Fahrgestellnummern (VIN)	VehicleIdentificationNumber	Fahrgestellnummer , niv,numarul de identificare , numarul seriei de sasiu,serie sasiu, numer VIN,Número de Identificação do Veículo, Número de Identificación de Automóviles ,numéro d'identification du véhicule,vehicle identification number,vin,VIN numerais	Beliebig	CloudWatch h Protokoll e können erkennen VINs , die aus einer Sequenz von 17 Zeichen bestehen und den Normen ISO 3779 und 3780 entsprech en. Diese Standards wurden für den weltweite n Einsatz konzipier t.

Schlüsselwörter für Identifikationsnummern des Führerscheins

Um verschiedene Arten von Führerschein-Identifikationsnummern zu erkennen, benötigt CloudWatch Logs ein Schlüsselwort, das sich in der Nähe der Nummern befindet. In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die CloudWatch Logs für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Australien	dl# dl:, dl :, dlno# driver licence, driver license, driver permit, drivers lic., drivers licence, driver's licence, drivers license, driver's license, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
Österreich	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik österreich
Belgien	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
Bulgarien	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
Kanada	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit,

Land oder Region	Schlüsselwörter
	drivers permit number, driving licence, driving license, driving permit, permis de conduire
Kroatien	vozačka dozvola
Zypern	άδεια οδήγησης
Tschechische Republik	číslo licence, číslo licence řidiče, číslo řidičskéh o průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský průkaz, řidičský průkaz
Dänemark	kørekort, kørekortnummer
Estland	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
Finnland	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
Frankreich	permis de conduire
Deutschland	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, fñhrerschein, fuhrerschein- nr, fñhrerschein- nr, fuhrersch einnummer, fñhrerscheinnnummer
Griechenland	δεια οδήγησης, adeia odigisis
Ungarn	illesztőprogramok lic, jogosítvány, jogsí, licensszám, vezető engedély, vezetői engedély
Irland	ceadúnas tiomána
Italien	patente di guida, patente di guida numero, patente guida, patente guida numero

Land oder Region	Schlüsselwörter
Lettland	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
Litauen	vairuotojo pažymėjimas
Luxemburg	fahrerlaubnis, führerschein
Malta	liċenzja tas-sewqan
Niederlande	permis de conduire, rijbewijs, rijbewijsnummer
Polen	numer licencyjny, prawo jazdy, zezwolenie na prowadzenie
Portugal	carta de condução, carteira de habilitação, carteira de motorist, carteira habilitação, carteira motorist, licença condução, licença de condução, número de licença, número licença, permissão condução, permissão de condução
Rumänien	numărul permisului de conducere, permis de conducere
Slowakei	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
Slowenien	vozniško dovoljenje

Land oder Region	Schlüsselwörter
Spanien	carnet conducir, el carnet de conducir, licencia conducir, licencia de manejo, número carnet conducir, número de carnet de conducir, número de permiso conducir, número de permiso de conducir, número licencia conducir, número permiso conducir, permiso conducción, permiso conducir, permiso de conducción
Schweden	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.
Großbritannien und Nordirland	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
Vereinigte Staaten	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit

Schlüsselwörter für nationale Identifikationsnummern

Um verschiedene Arten von nationalen Identifikationsnummern zu erkennen, benötigt CloudWatch Logs ein Schlüsselwort, das sich in unmittelbarer Nähe zu den Nummern befindet. Dazu gehören

Kennungen des Documento Nacional de Identidad (DNI) (Spanien), Codes des französischen Nationalen Instituts für Statistik und Wirtschaftsstudien (INSEE), deutsche Personalausweisnummern und Registro Geral (RG)-Nummern (Brasilien).

In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die CloudWatch Logs für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Brasilien	registro geral, rg
Frankreich	assurance sociale, carte nationale d'identité, cni, code sécurité sociale, French social security number, fssn#, insee, insurance number, national id number, nationalid#, numéro d'assurance, sécurité sociale, sécurité sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
Deutschland	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
Italien	codice fiscale, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
Spanien	dni, dni#, dninúmero#, documento nacional de identidad, identidad único, identidadúnico#, insurance number, national identification number, national identity, nationalid#, nationali dno#, número nacional identidad, personal identification number, personal identity no, unique identity number, uniqueid#

Schlüsselwörter für Passnummern

Um verschiedene Arten von Passnummern zu erkennen, benötigt CloudWatch Logs ein Schlüsselwort, das sich in der Nähe der Nummern befindet. In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die CloudWatch Logs für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Kanada	passport, passport#, passport, passport#, passportno, passportno#
Frankreich	numéro de passport, passport, passport #, passport #, passportn °, passport n °, passportNon, passport non
Deutschland	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass-nr, reisepassnummer
Italien	italian passport number, numéro passport, numéro passport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
Spanien	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
Großbritannien und Nordirland	passport #, passport n °, passportNon, passport non, passportn °, passport #, passport no, passport number, passport#, passportid
Vereinigte Staaten	passport, travel document

Schlüsselwörter für Steuerpflichtigenidentifikations- und Referenznummern

Um verschiedene Arten von Steueridentifikations- und Referenznummern zu erkennen, benötigt CloudWatch Logs ein Schlüsselwort, das sich in der Nähe der Nummern befindet. In der folgenden Tabelle sind die Schlüsselwörter aufgeführt, die CloudWatch Logs für bestimmte Länder und Regionen erkennt.

Land oder Region	Schlüsselwörter
Brasilien	cadastro de pessoa física, cadastro de pessoa física, cadastro de pessoas físicas, cadastro de pessoas físicas, cadastro nacional da pessoa jurídica, cadastro nacional da pessoa jurídica, cnpj, cpf
Frankreich	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#
Deutschland	identifikationsnummer, steuer id, steueridentifikationsnummer, steuernummer, tax id, tax identification number, tax number
Spanien	cif, cif número, cifnúmero#, nie, nif, número de contribuyente, número de identidad de extranjero, número de identificación fiscal, número de impuesto corporativo, personal tax number, tax id, tax identification number, tax number, tin, tin#
Großbritannien und Nordirland	paye, tax id, tax id no., tax id number, tax identification, tax identification#, tax no., tax number, tax reference, tax#, taxid#, temporary reference number, tin, trn, unique tax reference, unique taxpayer reference, utr
Vereinigte Staaten	Individuelle Steuerzahler-Identifikationsnummer (ITIN)

Daten-ID ARNs für persönlich identifizierbare Informationen (PII)

In der folgenden Tabelle sind die Amazon-Ressourcennamen (ARNs) für die Datenkennungen mit personenbezogenen Daten (PII) aufgeführt, die Sie zu Ihren Datenschutzrichtlinien hinzufügen können.

PII-Datenbezeichner ARNs

`arn:aws:dataprotection::aws:data-identifier/Address`

`arn:aws:dataprotection::aws:data-identifier/CepCode-BR`

`arn:aws:dataprotection::aws:data-identifier/Cnpj-BR`

`arn:aws:dataprotection::aws:data-identifier/CpfCode-BR`

`arn:aws:dataprotection::aws:data-identifier/DriversLicense-AT`

`arn:aws:dataprotection::aws:data-identifier/DriversLicense-AU`

`arn:aws:dataprotection::aws:data-identifier/DriversLicense-BE`

`arn:aws:dataprotection::aws:data-identifier/DriversLicense-BG`

`arn:aws:dataprotection::aws:data-identifier/DriversLicense-CA`

`arn:aws:dataprotection::aws:data-identifier/DriversLicense-CY`

`arn:aws:dataprotection::aws:data-identifier/DriversLicense-CZ`

`arn:aws:dataprotection::aws:data-identifier/DriversLicense-DE`

`arn:aws:dataprotection::aws:data-identifier/DriversLicense-DK`

`arn:aws:dataprotection::aws:data-identifier/DriversLicense-EE`

`arn:aws:dataprotection::aws:data-identifier/DriversLicense-ES`

`arn:aws:dataprotection::aws:data-identifier/DriversLicense-FI`

`arn:aws:dataprotection::aws:data-identifier/DriversLicense-FR`

PII-Datenbezeichner ARNs

arn:aws:dataprotection::aws:data-identifier/DriversLicense-GB

arn:aws:dataprotection::aws:data-identifier/DriversLicense-GR

arn:aws:dataprotection::aws:data-identifier/DriversLicense-HR

arn:aws:dataprotection::aws:data-identifier/DriversLicense-HU

arn:aws:dataprotection::aws:data-identifier/DriversLicense-IE

arn:aws:dataprotection::aws:data-identifier/DriversLicense-IT

arn:aws:dataprotection::aws:data-identifier/DriversLicense-LT

arn:aws:dataprotection::aws:data-identifier/DriversLicense-LU

arn:aws:dataprotection::aws:data-identifier/DriversLicense-LV

arn:aws:dataprotection::aws:data-identifier/DriversLicense-MT

arn:aws:dataprotection::aws:data-identifier/DriversLicense-NL

arn:aws:dataprotection::aws:data-identifier/DriversLicense-PL

arn:aws:dataprotection::aws:data-identifier/DriversLicense-PT

arn:aws:dataprotection::aws:data-identifier/DriversLicense-RO

arn:aws:dataprotection::aws:data-identifier/DriversLicense-SE

arn:aws:dataprotection::aws:data-identifier/DriversLicense-SI

arn:aws:dataprotection::aws:data-identifier/DriversLicense-SK

arn:aws:dataprotection::aws:data-identifier/DriversLicense-US

arn:aws:dataprotection::aws:data-identifier/ElectoralRollNumber-GB

arn:aws:dataprotection::aws:data-identifier/EmailAddress

PII-Datenbezeichner ARNs

arn:aws:dataprotection::aws:data-identifier/IndividualTaxIdentificationNumber-US

arn:aws:dataprotection::aws:data-identifier/InseeCode-FR

arn:aws:dataprotection::aws:data-identifier/LatLong

arn:aws:dataprotection::aws:data-identifier/Name

arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-DE

arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-ES

arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-IT

arn:aws:dataprotection::aws:data-identifier/NieNumber-ES

arn:aws:dataprotection::aws:data-identifier/NifNumber-ES

arn:aws:dataprotection::aws:data-identifier/PassportNumber-CA

arn:aws:dataprotection::aws:data-identifier/PassportNumber-DE

arn:aws:dataprotection::aws:data-identifier/PassportNumber-ES

arn:aws:dataprotection::aws:data-identifier/PassportNumber-FR

arn:aws:dataprotection::aws:data-identifier/PassportNumber-GB

arn:aws:dataprotection::aws:data-identifier/PassportNumber-IT

arn:aws:dataprotection::aws:data-identifier/PassportNumber-US

arn:aws:dataprotection::aws:data-identifier/PermanentResidenceNumber-CA

PII-Datenbezeichner ARNs

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-BR

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-DE

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-ES

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-FR

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-GB

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-IT

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US

arn:aws:dataprotection::aws:data-identifier/PostalCode-CA

arn:aws:dataprotection::aws:data-identifier/RgNumber-BR

arn:aws:dataprotection::aws:data-identifier/SocialInsuranceNumber-CA

arn:aws:dataprotection::aws:data-identifier/Ssn-ES

arn:aws:dataprotection::aws:data-identifier/Ssn-US

arn:aws:dataprotection::aws:data-identifier/TaxId-DE

arn:aws:dataprotection::aws:data-identifier/TaxId-ES

arn:aws:dataprotection::aws:data-identifier/TaxId-FR

arn:aws:dataprotection::aws:data-identifier/TaxId-GB

arn:aws:dataprotection::aws:data-identifier/VehicleIdentificationNumber

arn:aws:dataprotection::aws:data-identifier/ZipCode-US

Benutzerdefinierte Datenbezeichner

Themen

- [Was sind benutzerdefinierte SNS-Datenkennungen?](#)
- [Einschränkungen für benutzerdefinierte Datenkennungen](#)
- [Verwenden von benutzerdefinierten Datenbezeichnern in der Konsole](#)
- [Verwenden benutzerdefinierter Datenkennungen in Ihrer Datenschutzrichtlinie](#)

Was sind benutzerdefinierte SNS-Datenkennungen?

Mit benutzerdefinierten Datenbezeichnern (CDIs) können Sie Ihre eigenen benutzerdefinierten regulären Ausdrücke definieren, die in Ihrer Datenschutzrichtlinie verwendet werden können. Mithilfe von benutzerdefinierten Datenkennungen können Sie gezielt auf geschäftsspezifische Anwendungsfälle mit persönlich identifizierbaren Informationen (PII) abzielen, die [verwaltete Datenkennungen](#) nicht bieten können. Sie können beispielsweise eine benutzerdefinierte Daten-ID verwenden, um nach einem unternehmensspezifischen Mitarbeiter zu suchen. IDs Benutzerdefinierte Datenkennungen können in Verbindung mit verwalteten Datenkennungen verwendet werden.

Einschränkungen für benutzerdefinierte Datenkennungen

CloudWatch Für benutzerdefinierte Datenkennungen von Protokollen gelten die folgenden Einschränkungen:

- Datenschutzrichtlinien unterstützen derzeit maximal 10 benutzerdefinierte Datenkennungen.
- Die Namen von benutzerdefinierten Datenkennungen sind auf maximal 128 Zeichen beschränkt. Folgende Zeichen werden unterstützt:
 - Alphanumerisch: (a-zA-Z0-9)
 - Sonderzeichen: („_“ | „-“)
- RegEx ist auf maximal 200 Zeichen beschränkt. Folgende Zeichen werden unterstützt:
 - Alphanumerisch: (a-zA-Z0-9)
 - Sonderzeichen: („_“ | „#“ | „=“ | „@“ | „/“ | „.“ | „“ | „-“ | „ “)
 - RegEx reservierte Zeichen: („^“ | „\$“ | „?“ | „[“ | „]“ | „{“ | „}“ | „|“ | „\“ | „*“ | „+“ | „.“)
- Benutzerdefinierte Datenkennungen können nicht denselben Namen wie verwaltete Datenkennungen haben.

- Benutzerdefinierte Datenbezeichner können in einer Datenschutzrichtlinie auf Kontoebene oder in Datenschutzrichtlinien auf Protokollgruppenebene angegeben werden. Ähnlich wie verwaltete Datenkennungen funktionieren benutzerdefinierte Datenkennungen, die in einer Richtlinie auf Kontoebene definiert sind, in Kombination mit benutzerdefinierten Datenkennungen, die in einer Richtlinie auf Protokollgruppenebene definiert sind.

Verwenden von benutzerdefinierten Datenbezeichnern in der Konsole

Wenn Sie die CloudWatch Konsole verwenden, um eine Datenschutzrichtlinie zu erstellen oder zu bearbeiten, geben Sie zur Angabe einer benutzerdefinierten Daten-ID einfach einen Namen und einen regulären Ausdruck für die Daten-ID ein. Sie können dies beispielsweise **Employee_ID** für den Namen und **EmployeeID-\d{9}** als regulären Ausdruck eingeben. Dieser reguläre Ausdruck erkennt und maskiert Protokollereignisse mit neun nachfolgenden Zahlen EmployeeID-. Beispiel: EmployeeID-123456789

Verwenden benutzerdefinierter Datenkennungen in Ihrer Datenschutzrichtlinie

Wenn Sie die AWS API AWS CLI oder verwenden, um eine benutzerdefinierte Daten-ID anzugeben, müssen Sie den Namen der Daten-ID und den regulären Ausdruck in die JSON-Richtlinie aufnehmen, die zur Definition der Datenschutzrichtlinie verwendet wird. Die folgende Datenschutzrichtlinie erkennt und maskiert Protokollereignisse, die unternehmensspezifische Mitarbeiter betreffen. IDs

1. Erstellen Sie einen Configuration-Block innerhalb Ihrer Datenschutzrichtlinie.
2. Geben Sie eine Name als benutzerdefinierte Datenkennung ein. Beispiel, **EmployeeId**.
3. Geben Sie eine Regex als benutzerdefinierte Datenkennung ein. Beispiel, **EmployeeID-\d{9}**. Dieser reguläre Ausdruck entspricht Protokollereignissen EmployeeID-, die neun Ziffern dahinter enthalten. EmployeeID- Beispiel: EmployeeID-123456789
4. Beziehen Sie sich in einer Richtlinienerklärung auf die folgende benutzerdefinierte Datenkennung.

```
{
  "Name": "example_data_protection_policy",
  "Description": "Example data protection policy with custom data identifiers",
  "Version": "2021-06-01",
  "Configuration": {
    "CustomDataIdentifier": [
      {"Name": "EmployeeId", "Regex": "EmployeeId-\\d{9}"}
    ]
  }
}
```

```

    ]
  },
  "Statement": [
    {
      "Sid": "audit-policy",
      "DataIdentifier": [
        "EmployeeId"
      ],
      "Operation": {
        "Audit": {
          "FindingsDestination": {
            "S3": {
              "Bucket": "EXISTING_BUCKET"
            }
          }
        }
      }
    },
    {
      "Sid": "redact-policy",
      "DataIdentifier": [
        "EmployeeId"
      ],
      "Operation": {
        "Deidentify": {
          "MaskConfig": {
            }
          }
        }
      }
    }
  ]
}

```

5. (Optional) Fügen Sie dem Configuration-Block bei Bedarf weitere benutzerdefinierte Datenkennungen hinzu. Datenschutzrichtlinien unterstützen derzeit maximal 10 benutzerdefinierte Datenkennungen.

Logs während der Aufnahme transformieren

Mit der Transformation und Anreicherung von Protokollen können Sie alle Ihre Protokolle zum Zeitpunkt der Aufnahme in Logs in ein konsistentes und kontextreiches Format normalisieren. CloudWatch [Sie können Ihren Protokollen Struktur verleihen, indem Sie vorkonfigurierte Vorlagen für gängige AWS Dienste wie AWS WAF Amazon Route 53 verwenden oder benutzerdefinierte Transformatoren mit systemeigenen Parsern wie Grok erstellen](#). Sie können auch bestehende Attribute umbenennen und Ihren Protokollen zusätzliche Metadaten wie Konto-ID und Region hinzufügen.

Die Protokolltransformation trägt dazu bei, Ihre Protokollabfragen in Ihren Anwendungen zu vereinfachen und zu verkürzen und die Erstellung von Warnmeldungen in Ihren Protokollen zu vereinfachen. Diese Funktion ermöglicht die Transformation gängiger Protokolltypen mit out-of-the-box Transformationsvorlagen für wichtige AWS Protokollquellen wie VPC Flow-Logs, Route 53 und Amazon RDS for PostgreSQL. Sie können vorkonfigurierte Transformationsvorlagen verwenden oder benutzerdefinierte Transformatoren erstellen, die Ihren Anforderungen entsprechen.

Die Protokolltransformation hilft Ihnen bei der Verwaltung von Protokollen, die aus verschiedenen Quellen stammen und sich in Format und Attributnamen stark unterscheiden.

Nachdem Sie einen Transformer erstellt haben, werden die aufgenommenen Protokollereignisse konvertiert und in einem Standardformat gespeichert. Sie können diese transformierten Logs nutzen, um Ihre Analytics-Erfahrung mit den folgenden Funktionen zu beschleunigen:

- [Feldindizes](#)
- [CloudWatch Logs Insights entdeckte Felder](#)
- Flexibilität bei der Alarmierung mithilfe [metrischer Filter](#)
- Weiterleitung über [Abonnementfilter](#)
- Erstellen von Metrikdaten aus Protokollereignissen mit [Contributor Insights](#), wobei Sie wählen können, ob die Contributor Insights-Regel Protokollereignisse entweder vor oder nach ihrer Transformation auswertet.

Transformationen finden nur während der Protokollaufnahme statt. Sie können Protokollereignisse, die bereits aufgenommen wurden, nicht transformieren. Transformationen sind nicht reversibel. Sowohl ursprüngliche als auch transformierte Protokolle werden in CloudWatch Protokollen mit derselben Aufbewahrungsrichtlinie gespeichert. Die Fähigkeit zur Umwandlung und Anreicherung von

Protokollen ist im aktuellen Aufnahmepreis der Standardprotokollklasse enthalten. Die Kosten für die Speicherung von Protokollen richten sich nach der Protokollgröße nach der Transformation, die das ursprüngliche Protokollvolumen übersteigen kann.

 Important

Nachdem die Protokollereignisse transformiert wurden, müssen Sie CloudWatch Logs Insights-Abfragen verwenden, um die transformierten Versionen der Protokolle einzusehen. Die [FilterLogEvents](#)Aktionen [GetLogEvents](#) und geben nur die Originalversionen der Protokollereignisse zurück, bevor sie transformiert wurden.

 Important

Obwohl ein einzelnes Protokollereignis bis zu 1 MB PutLogEvents erlaubt, kann die Protokolltransformation nur Protokollereignisse mit einer Größe von weniger als 512 KB verarbeiten. Alle Protokollereignisse, die größer als 512 KB sind, schlagen bei der Transformation fehl und geben einen Fehler aus. Die Gesamtgröße von PutLogEvents kann immer noch 512 KB überschreiten.

Neben der Umwandlung in verschiedene Formate können Sie Ihre Logs auch mit zusätzlichem Kontext wie Konto-ID, Region und Schlüsselwort anreichern. Diese werden aus dem Namen der Protokollgruppe und aus statischen Schlüsselwörtern extrahiert.

Die Protokolltransformation hilft Ihnen bei Protokollen, die aus verschiedenen Quellen stammen und sich in Format und Attributnamen stark unterscheiden.

Die Transformation und Anreicherung von Protokollen wird nur für Protokollgruppen der Standard-Protokollklasse unterstützt.

Sie können Transformatoren für einzelne Protokollgruppen erstellen, und Sie können auch Transformatoren auf Kontoebene erstellen, die für alle oder viele Protokollgruppen in Ihrem Konto gelten. Wenn eine Protokollgruppe über einen Transformator auf Protokollgruppenebene verfügt, hat dieser Transformator Vorrang vor allen Transformatoren auf Kontoebene, die andernfalls für diese Protokollgruppe gelten würden.

Themen

- [Log-Transformatoren erstellen und verwalten](#)

- [Konfigurierbare Prozessoren vom Typ Parser](#)
- [Integrierte Prozessoren für AWS verkaufte Logs](#)
- [Prozessoren mutieren durch Zeichenketten](#)
- [JSON-Prozessoren mutieren](#)
- [Prozessoren für Datentypkonverter](#)
- [Metriken und Fehler bei der Transformation](#)

Log-Transformatoren erstellen und verwalten

Ein Protokolltransformator umfasst einen oder mehrere Prozessoren, die sich in einer logischen Pipeline befinden. Jeder Prozessor wird nacheinander in der Reihenfolge, in der er in der Transformer-Konfiguration aufgeführt ist, auf ein Protokollereignis angewendet.

Einige Prozessoren sind vom Typ Parser. Jeder Transformator muss mindestens einen Parser haben, und der erste Prozessor in einem Transformator muss ein Parser sein.

Bei einigen der Parser handelt es sich um integrierte Parser, die für einen bestimmten Typ von versendeten Protokollen konfiguriert sind. AWS

Andere Prozessortypen sind String-Mutatoren, JSON-Mutatoren und Datenprozessoren.

Sie können Transformatoren für einzelne Protokollgruppen erstellen, und Sie können auch Transformatoren auf Kontoebene erstellen, die für alle oder viele Protokollgruppen in Ihrem Konto gelten. Wenn eine Protokollgruppe über einen Transformator auf Protokollgruppenebene verfügt, hat dieser Transformator Vorrang vor allen Transformatoren auf Kontoebene, die andernfalls für diese Protokollgruppe gelten würden. Sie können bis zu 20 Transformatoren auf Kontoebene in einer Region in Ihrem Konto haben.

Bei der Erstellung eines Transformators müssen Sie die folgenden Richtlinien beachten:

- Wenn Sie einen vorkonfigurierten Parser für einen bestimmten Typ von AWS versendeten Protokollen verwenden, muss es sich um den ersten Prozessor handeln, der im Transformer aufgeführt ist. Sie können nur einen solchen Prozessor in einen Transformator aufnehmen.
- Sie können nur einen grok Prozessor in einen Transformator aufnehmen.
- In einem Transformator muss mindestens ein Prozessor vom Typ Parser vorhanden sein. Sie können bis zu fünf Prozessoren vom Typ Parser einbeziehen. Diese Grenze von fünf umfasst sowohl integrierte Parser als auch konfigurierbare Parser.

- Sie können bis zu 20 Prozessoren in einem Transformator haben.
- Sie können nur einen AddKeys-Prozessor in einen Transformer aufnehmen.
- Sie können nur einen CopyValue-Prozessor in einen Transformer aufnehmen.
- Jeder Transformator kann bis zu 200 Felder aus einem Protokollereignis extrahieren.
- Jedes Protokollereignis MUSS unter 512 KB liegen. Die Gesamtgröße der Protokollereignisse kann immer noch 512 KB überschreiten.

Themen

- [Erstellen Sie eine Transformer-Richtlinie auf Kontoebene](#)
- [Bearbeiten oder löschen Sie eine Transformer-Richtlinie auf Kontoebene](#)
- [Erstellen Sie einen log-group-level Log-Transformer von Grund auf](#)
- [Erstellen Sie einen log-group-level Transformator, indem Sie einen vorhandenen kopieren](#)
- [Bearbeiten Sie einen log-group-level Transformator](#)
- [Löschen Sie einen log-group-level Transformator](#)

Erstellen Sie eine Transformer-Richtlinie auf Kontoebene

Verwenden Sie die Schritte in diesem Abschnitt, um eine Transformer-Richtlinie zu erstellen, die für alle Protokollgruppen im Konto oder für mehrere Protokollgruppen gilt, deren Protokollgruppennamen mit derselben Zeichenfolge (Präfix) beginnen. In einer Region können bis zu 20 Transformer-Richtlinien auf Kontoebene eingerichtet werden.

Sie können nicht zwei Transformer-Richtlinien in derselben Region erstellen, die dasselbe Präfix verwenden oder bei denen ein Präfix in einem anderen enthalten ist. Wenn Sie beispielsweise eine Transformer-Richtlinie für das Zeichenfolgenpräfix erstellen `/aws/lambda`, können Sie keine weitere mit dem Präfix erstellen `/aws`. Aber Sie könnten einen Transformator für `/aws/lambda` und einen anderen für `/aws/waf` haben.

Um eine Transformer-Richtlinie auf Kontoebene zu erstellen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im linken Navigationsbereich Einstellungen und dann die Registerkarte Protokolle aus.
3. Wählen Sie im Abschnitt Transformer-Richtlinie für Konto die Option Transformer-Richtlinie erstellen aus.

4. Geben Sie unter Transformer-Richtliniename einen Namen für Ihre neue Richtlinie ein.
5. Führen Sie für Select log groups einen der folgenden Schritte aus:
 - Wählen Sie Alle Standard-Protokollgruppen, damit die Transformer-Richtlinie auf alle Standardklassen-Protokollgruppen im Konto angewendet wird.
 - Wählen Sie Protokollgruppen nach Präfixübereinstimmung, um die Richtlinie auf eine Teilmenge von Protokollgruppen anzuwenden, deren Namen alle mit derselben Zeichenfolge beginnen. Geben Sie dann das Präfix für diese Protokollgruppen unter Auswahlkriterien ein.
6. Wählen Sie im Bereich „Parser auswählen“ mithilfe von Parsern einen Parser aus, der in Ihren Transformer aufgenommen werden soll.

Handelt es sich um einen vorkonfigurierten Parser für einen bestimmten AWS Protokolltyp, müssen Sie dafür keine Konfiguration angeben.

Wenn es sich um einen anderen Parser handelt, müssen Sie dessen Konfiguration angeben. Weitere Informationen finden Sie in den Informationen zu diesem Prozessor unter [Konfigurierbare Prozessoren vom Typ Parser](#).

7. Um einen weiteren Prozessor hinzuzufügen, wählen Sie Prozessor auswählen. Wählen Sie dann im Feld Prozessor den gewünschten Prozessor aus und geben Sie die Konfigurationsparameter ein.

Denken Sie daran, dass Prozessoren die Protokollereignisse in der Reihenfolge verarbeiten, in der Sie sie dem Transformator hinzufügen.

8. (Optional) Um weitere Prozessoren hinzuzufügen, wählen Sie + Prozessor und wiederholen Sie den vorherigen Schritt.
9. (Optional) Sie können den Transformator, den Sie bisher erstellt haben, jederzeit anhand eines Beispielprotokollereignisses testen. Führen Sie dazu im Abschnitt Transformer-Vorschau einen der folgenden Schritte aus:
 - Wählen Sie unter Protokollgruppen auswählen bis zu fünf Protokollgruppen aus und wählen Sie dann Letzte Protokollereignisse laden aus. Wählen Sie dann Transformer testen.
 - Kopieren Sie Protokollereignisse direkt in Beispielprotokollereignisse und wählen Sie dann Test Transformer.

Die transformierte Version des Protokolls wird dann angezeigt.

10. Wenn Sie mit dem Hinzufügen von Prozessoren fertig sind und mit den Tests der Beispielprotokolle zufrieden sind, wählen Sie Speichern.

11. Klicken Sie danach auf Erstellen.

Bearbeiten oder löschen Sie eine Transformer-Richtlinie auf Kontoebene

Gehen Sie wie in diesem Abschnitt beschrieben vor, um eine Transformer-Richtlinie auf Kontoebene zu bearbeiten oder zu löschen.

Um eine Transformer-Richtlinie auf Kontoebene zu bearbeiten oder zu löschen

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im linken Navigationsbereich Einstellungen und dann die Registerkarte Protokolle aus.
3. Wählen Sie im Abschnitt Transformer-Kontorichtlinie die Option Verwalten aus.
4. Wählen Sie die Schaltfläche neben der Transformer-Richtlinie aus, die Sie verwalten möchten, und wählen Sie dann Bearbeiten oder Löschen.

Wenn Sie die Richtlinie bearbeiten, finden Sie in [Konfigurierbare Prozessoren vom Typ Parser](#) den Schritten 5-11 unter Ihre Optionen.

Erstellen Sie einen log-group-level Log-Transformer von Grund auf

Gehen Sie wie folgt vor, um einen log-group-level Transformator von Grund auf neu zu erstellen.

So verwenden Sie die Konsole, um einen Protokolltransformator für eine Protokollgruppe zu erstellen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle), Log groups (Protokollgruppen) aus.
3. Wählen Sie die Protokollgruppe aus, für die Sie den Transformer erstellen möchten.
4. Wählen Sie die Registerkarte Transformer. Möglicherweise müssen Sie in der Tab-Liste nach rechts scrollen, um sie zu sehen.
5. Wählen Sie Create Transformer.
6. Wählen Sie im Feld Wählen Sie einen Parser aus, der in Ihren Transformer aufgenommen werden soll.

Wenn es sich um einen vorkonfigurierten Parser für einen Typ von AWS verkauftem Protokoll handelt, müssen Sie dafür keine Konfiguration angeben.

Wenn es sich um einen anderen Parser handelt, müssen Sie dessen Konfiguration angeben. Weitere Informationen finden Sie in den Informationen zu diesem Prozessor unter [Konfigurierbare Prozessoren vom Typ Parser](#).

7. Um einen weiteren Prozessor hinzuzufügen, wählen Sie + Prozessor hinzufügen. Wählen Sie dann im Feld Prozessoren auswählen den gewünschten Prozessor aus und geben Sie die Konfigurationsparameter ein.

Denken Sie daran, dass Prozessoren die Protokollereignisse in der Reihenfolge verarbeiten, in der Sie sie dem Transformator hinzufügen.

8. (Optional) Sie können den Transformator, den Sie bisher erstellt haben, jederzeit anhand eines Beispielprotokollereignisses testen. Gehen Sie dazu wie folgt vor:
 - Wählen Sie im Abschnitt Transformationsvorschau entweder Beispielprotokoll laden aus, um ein Beispielprotokollereignis aus der Protokollgruppe zu laden, für die dieser Transformator bestimmt ist, oder fügen Sie ein Protokollereignis in das Textfeld ein.

Wählen Sie Transformator testen aus. Die transformierte Version des Protokolls wird angezeigt

9. Wenn Sie mit dem Hinzufügen von Prozessoren fertig sind und mit den Tests der Beispielprotokolle zufrieden sind, wählen Sie Speichern.

Um den zu verwenden AWS CLI , um einen Log-Transformator von Grund auf neu zu erstellen

- Verwenden Sie den Befehl `aws logs put-transformer`. Wenn Sie `parseJSON` es als ersten Prozessor verwenden, müssen Sie das gesamte Protokollereignis `@message` als Quellfeld analysieren. Nach der ersten JSON-Analyse können Sie dann bestimmte Felder in nachfolgenden Prozessoren bearbeiten. Im Folgenden finden Sie ein Beispiel, das einen Transformator erstellt, der die `addKeys` Prozessoren `parseJSON` und enthält:

```
aws logs put-transformer \
  --transformer-config '[{"parseJSON":{"source":"@message"}}, {"addKeys":
{"entries":[{"key":"metadata.transformed_in", "value":"CloudWatchLogs"},
{"key":"feature", "value":"Transformation"}]}], {"trimString":{"withKeys":
["status"]}]}' \
  --log-group-identifier my-log-group-name
```

Erstellen Sie einen log-group-level Transformator, indem Sie einen vorhandenen kopieren

Sie können die Konsole verwenden, um die JSON-Konfiguration eines vorhandenen Transformators zu kopieren. Sie können diesen Code dann verwenden, um einen identischen Transformator zu erstellen, indem Sie den verwenden AWS CLI, oder Sie können zuerst die Konfiguration ändern.

Um einen Log-Transformator zu erstellen, indem Sie einen vorhandenen kopieren

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle), Log groups (Protokollgruppen) aus.
3. Wählen Sie die Protokollgruppe aus, die den Transformator enthält, den Sie kopieren möchten.
4. Wählen Sie die Registerkarte Transformationen. Möglicherweise müssen Sie in der Tab-Liste nach rechts scrollen, um sie zu sehen.
5. Wählen Sie Transformer verwalten.
6. Wählen Sie Transformator kopieren. Dadurch wird der Transformer-JSON in Ihre Zwischenablage kopiert.
7. Erstellen Sie eine Datei und fügen Sie sie in die Transformer-Konfiguration ein. In diesem Beispiel rufen wir die Datei auf CopiedTransformer.json
8. Verwenden Sie den AWS CLI , um einen neuen Transformator mit dieser Konfiguration zu erstellen.

```
aws logs put-transformer --log-group-identifier my-log-group-name \  
--transformer-config file://CopiedTransformer.json
```

Bearbeiten Sie einen log-group-level Transformator

Gehen Sie wie folgt vor, um einen vorhandenen Log-Transformer zu bearbeiten.

Um einen Log-Transformer zu bearbeiten

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle), Log groups (Protokollgruppen) aus.
3. Wählen Sie die Protokollgruppe aus, die den Transformator enthält, den Sie bearbeiten möchten.

4. Wählen Sie die Registerkarte Transformationen. Möglicherweise müssen Sie in der Tab-Liste nach rechts scrollen, um sie zu sehen.
5. Wählen Sie Transformer verwalten.
6. Nehmen Sie in den Abschnitten Parser und Prozessoren Ihre Änderungen vor.
7. Um einen weiteren Prozessor hinzuzufügen, wählen Sie + Prozessor hinzufügen. Wählen Sie dann im Feld Prozessor den gewünschten Prozessor aus und geben Sie die Konfigurationsparameter ein.

Denken Sie daran, dass Prozessoren die Protokollereignisse in der Reihenfolge verarbeiten, in der Sie sie dem Transformator hinzufügen.

8. (Optional) Sie können den Transformator, den Sie bisher erstellt haben, jederzeit anhand eines Beispielprotokollereignisses testen. Gehen Sie dazu wie folgt vor:
 - Wählen Sie im Abschnitt Transformationsvorschau entweder Beispielprotokoll laden aus, um ein Beispielprotokollereignis aus der Protokollgruppe zu laden, für die dieser Transformator bestimmt ist, oder fügen Sie ein Protokollereignis in das Textfeld ein.

Wählen Sie „Transformation testen“. Die transformierte Version des Protokolls wird angezeigt

9. Wenn Sie mit dem Hinzufügen von Prozessoren fertig sind und mit den Tests der Beispielprotokolle zufrieden sind, wählen Sie Speichern.

Löschen Sie einen log-group-level Transformator

Gehen Sie wie folgt vor, um einen Protokolltransformator zu löschen.

Um einen Log-Transformer zu löschen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle), Log groups (Protokollgruppen) aus.
3. Wählen Sie die Protokollgruppe aus, die den Transformator enthält, den Sie bearbeiten möchten.
4. Wählen Sie die Registerkarte Transformationen. Möglicherweise müssen Sie in der Tab-Liste nach rechts scrollen, um sie zu sehen.
5. Wählen Sie Löschen aus.
6. Wählen Sie im Bestätigungsfeld die Option Richtlinie löschen aus.

Konfigurierbare Prozessoren vom Typ Parser

Dieser Abschnitt enthält Informationen zu den konfigurierbaren Datenparser-Prozessoren, die Sie in einem Log-Event-Transformator verwenden können.

Inhalt

- [parseJSON](#)
- [Grok](#)
 - [Grok-Beispiele](#)
 - [Beispiel 1: Verwenden Sie Grok, um ein Feld aus unstrukturierten Protokollen zu extrahieren](#)
 - [Beispiel 2: Verwenden Sie grok in Kombination mit parseJSON, um Felder aus einem JSON-Protokollereignis zu extrahieren](#)
 - [Beispiel 3: Grok-Muster mit gepunkteter Anmerkung in FIELD_NAME](#)
 - [Unterstützte Grok-Muster](#)
 - [Beispiele für gängige Protokollformate](#)
 - [Beispiel für ein Apache-Protokoll](#)
 - [Beispiel für ein NGINX-Protokoll](#)
 - [Beispiel für ein Syslog-Protokoll \(RFC 5424\)](#)
- [csv](#)
- [parseKeyValue](#)

parseJSON

Der ParseJSON-Prozessor analysiert JSON-Protokollereignisse und fügt extrahierte JSON-Schlüssel-Wert-Paare unter dem Ziel ein. Wenn Sie kein Ziel angeben, platziert der Prozessor das Schlüssel-Wert-Paar unter dem Stammknoten. Wenn Sie parseJSON es als ersten Prozessor verwenden, müssen Sie das gesamte Protokollereignis @message als Quellfeld analysieren. Nach der ersten JSON-Analyse können Sie dann bestimmte Felder in nachfolgenden Prozessoren bearbeiten.

Der ursprüngliche @message Inhalt wird nicht geändert, die neuen Schlüssel werden der Nachricht hinzugefügt.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Quelle	Pfad zu dem Feld im Protokollereignis, das analysiert wird. Verwenden Sie die Punktnotation, um auf untergeordnete Felder zuzugreifen. Beispiel: <code>store.book</code>	Nein	@message	Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
Ziel	Das Zielfeld der analysierten JSON-Datei	Nein	Parent JSON node	Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3

Beispiel

Angenommen, ein aufgenommenes Protokollereignis sieht wie folgt aus:

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

Wenn wir dann diesen ParseJSON-Prozessor haben:

```
[
  {
    "parseJSON": {
      "destination": "new_key"
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
```

```

    "new_key": {
      "outer_key": {
        "inner_key": "inner_value"
      }
    }
  }
}

```

Grok

Verwenden Sie den Grok-Prozessor, um unstrukturierte Daten mithilfe von Musterabgleich zu analysieren und zu strukturieren. Dieser Prozessor kann auch Felder aus Protokollnachrichten extrahieren.

Feld	Description	Erforderlich?	Standard	Einschränkungen	Hinweise
Quelle	Pfad des Felds, auf das der Grok-Matching angewendet werden soll	Nein	@message	Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3	
match	Das Grok-Muster, das mit dem Protokollereignis verglichen werden soll	Ja		Maximale Länge: 512 Maximale Anzahl von Grok-Mustern: 20 Für einige Grok-Mustertypen gelten individuelle Nutzungsbeschränkungen. Jede Kombination der folgenden Muster kann bis zu fünfmal verwendet werden: {URI,	Alle unterstützen Grok-Muster anzeigen

Feld	Description	Erforderlich?	Standard	Einschränkungen	Hinweise
				URIPARAM, URIPATHPARAM, SPACE, DATA, GREEDYDATA, GREEDYDATA_MULTILINE} Grok-Muster unterstützen keine Typkonvertierungen. Für gängige Protokollformatmuster (APACHE_ACCESS_LOG, NGINX_ACCESS_LOG, SYSLOG5424) werden nur die Muster DATA, GREEDYDATA oder GREEDYDATA_MULTILINE unterstützt, sodass sie nach dem allgemeinen Protokollmuster eingefügt werden können.	

Struktur eines Grok-Musters

Dies ist die unterstützte Grok-Musterstruktur:


```
%{PATTERN_NAME:FIELD_NAME}
```

- **PATTERN_NAME:** Bezieht sich auf einen vordefinierten regulären Ausdruck für den Abgleich mit einem bestimmten Datentyp. Es werden nur vordefinierte [Grok-Muster unterstützt](#). Das Erstellen von benutzerdefinierten Mustern ist nicht zulässig.
- **FIELD_NAME:** Weist dem extrahierten Wert einen Namen zu. **FIELD_NAME** ist optional, aber wenn Sie diesen Wert nicht angeben, werden die extrahierten Daten aus dem transformierten Protokollereignis gelöscht. Wenn die Notation mit Punkten **FIELD_NAME** verwendet wird (z. B. „parent.child“), wird dies als JSON-Pfad betrachtet.
- **Typkonvertierung:** Explizite Typkonvertierungen werden nicht unterstützt. Verwenden Sie den [TypeConverter Prozessor](#), um den Datentyp eines beliebigen von Grok extrahierten Werts zu konvertieren.

Um komplexere passende Ausdrücke zu erstellen, können Sie mehrere Grok-Muster kombinieren. Bis zu 20 Grok-Muster können kombiniert werden, um einem Protokollereignis zu entsprechen. Diese Kombination von Mustern `%{NUMBER:timestamp} [%{NUMBER:db} %{IP:client_ip}: %{NUMBER:client_port}] %{GREEDYDATA:data}` kann beispielsweise verwendet werden, um Felder aus einem langsamen Redis-Protokolleintrag wie folgt zu extrahieren:

```
1629860738.123456 [0 127.0.0.1:6379] "SET" "key1" "value1"
```

Grok-Beispiele

Beispiel 1: Verwenden Sie Grok, um ein Feld aus unstrukturierten Protokollen zu extrahieren

Beispielprotokoll:

```
293750 server-01.internal-network.local OK "[Thread-000] token generated"
```

Verwendeter Transformator:

```
[
  {
    "grok": {
      "match": "%{NUMBER:version} %{HOSTNAME:hostname} %{NOTSPACE:status}
%{QUOTEDSTRING:logMsg}"
    }
  }
]
```

```
]
```

Ausgabe:

```
{
  "version": "293750",
  "hostname": "server-01.internal-network.local",
  "status": "OK",
  "logMsg": "[Thread-000] token generated"
}
```

Beispielprotokoll:

```
23/Nov/2024:10:25:15 -0900 172.16.0.1 200
```

Verwendeter Transformator:

```
[
  {
    "grok": {
      "match": "%{HTTPDATE:timestamp} %{IPORHOST:clientip}
%{NUMBER:response_status}"
    }
  }
]
```

Ausgabe:

```
{
  "timestamp": "23/Nov/2024:10:25:15 -0900",
  "clientip": "172.16.0.1",
  "response_status": "200"
}
```

Beispiel 2: Verwenden Sie grok in Kombination mit parseJSON, um Felder aus einem JSON-Protokollereignis zu extrahieren

Beispielprotokoll:

```
{
```

```

    "timestamp": "2024-11-23T16:03:12Z",
    "level": "ERROR",
    "logMsg": "GET /page.html HTTP/1.1"
  }

```

Verwendeter Transformator:

```

[
  {
    "parseJSON": {}
  },
  {
    "grok": {
      "source": "logMsg",
      "match": "%{WORD:http_method} %{NOTSPACE:request} HTTP/"
    }
  }
]

```

Ausgabe:

```

{
  "timestamp": "2024-11-23T16:03:12Z",
  "level": "ERROR",
  "logMsg": "GET /page.html HTTP/1.1",
  "http_method": "GET",
  "request": "/page.html",
  "http_version": "1.1"
}

```

Beispiel 3: Grok-Muster mit gepunkteter Anmerkung in FIELD_NAME

Beispielprotokoll:

```
192.168.1.1 GET /index.html?param=value 200 1234
```

Verwendeter Transformator:

```

[
  {

```

```
"grok": {
  "match": "%{IP:client.ip} %{WORD:method} %{URIPATHPARAM:request.uri}
%{NUMBER:response.status} %{NUMBER:response.bytes}"
}
]
```

Ausgabe:

```
{
  "client": {
    "ip": "192.168.1.1"
  },
  "method": "GET",
  "request": {
    "uri": "/index.html?param=value"
  },
  "response": {
    "status": "200",
    "bytes": "1234"
  }
}
```

Unterstützte Grok-Muster

In den folgenden Tabellen sind die Muster aufgeführt, die vom grok Prozessor unterstützt werden.

Allgemeine Grok-Muster

Grok-Muster	Description	Maximales Musterlimit	Beispiel
BENUTZERN AME oder BENUTZER	Entspricht einem oder mehreren Zeichen, die Kleinbuchstaben (a-z), Großbuchstaben (A-Z), Ziffern (0-9), Punkte (.), Unterstriche (_) oder Bindestriche (-) enthalten können	20	Eingabe: user123.name-TEST Pattern: %{USERNAME:name} Ausgang: {"name": "user123.name-TEST"}

Grok-Muster	Description	Maximales Musterlimit	Beispiel
INT	Entspricht einem optionalen Plus- oder Minuszeichen, gefolgt von einer oder mehreren Ziffern.	20	Eingabe: -456 Pattern: <code>%{INT:num}</code> Ausgang: <code>{"num": "-456"}</code>
BASE10NUM	Entspricht einer Ganzzahl oder einer Gleitkommazahl mit optionalem Vorzeichen und Dezimaltrennzeichen	20	Eingabe: -0.67 Pattern: <code>%{BASE10NUM:num}</code> Ausgang: <code>{"num": "-0.67"}</code>
BASE16NUM	Entspricht Dezimal- und Hexadezimalzahlen mit einem optionalen Zeichen (+ oder -) und einem optionalen 0x-Präfix	20	Eingabe: +0xA1B2 Pattern: <code>%{BASE16NUM:num}</code> Ausgang: <code>{"num": "+0xA1B2"}</code>
POSEN	Entspricht ganzen positiven Ganzzahlen ohne führende Nullen, die aus einer oder mehreren Ziffern bestehen (1-9 gefolgt von 0-9)	20	Eingabe: 123 Pattern: <code>%{POSINT:num}</code> Ausgang: <code>{"num": "123"}</code>
NICHT NEGATIV	Entspricht allen ganzen Zahlen (bestehend aus einer oder mehreren Ziffern 0-9), einschließlich Null und Zahlen mit führenden Nullen.	20	Eingabe: 007 Pattern: <code>%{NONNEGINT:num}</code> Ausgang: <code>{"num": "007"}</code>

Grok-Muster	Description	Maximales Musterlimit	Beispiel
WORT	Entspricht ganzen Wörtern, die aus einem oder mehreren Wortzeichen (\ w) bestehen, einschließlich Buchstaben, Ziffern und Unterstrichen	20	Eingabe: user_123 Pattern: <code>%{WORD:user}</code> Ausgang: <code>{"user": "user_123"}</code>
KEIN LEERZEICHEN	Entspricht einem oder mehreren Zeichen, die keine Leerzeichen sind.	5	Eingabe: hello_world123 Pattern: <code>%{NOTSPACE:msg}</code> Ausgang: <code>{"msg": "hello_world123"}</code>
SPACE	Entspricht null oder mehr Leerzeichen.	5	Eingabe: " " Pattern: <code>%{SPACE:extra}</code> Ausgang: <code>{"extra": " "}</code>
DATA	Entspricht einem beliebigen Zeichen (außer Zeilenumbruch) null oder öfter, nicht gierig.	5	Eingabe: abc def ghi Pattern: <code>%{DATA:x}</code> <code>%{DATA:y}</code> Ausgang: <code>{"x": "abc", "y": "def ghi"}</code>
GIERIGE DATEN	Entspricht einem beliebigen Zeichen (außer Newline) null Mal oder öfter, gierig.	5	Eingabe: abc def ghi Pattern: <code>%{GREEDYDATA:x}</code> <code>%{GREEDYDATA:y}</code> Ausgang: <code>{"x": "abc def", "y": "ghi"}</code>

Grok-Muster	Description	Maximales Musterlimit	Beispiel
GREEDYDATA_MULTILINE	Entspricht einem beliebigen Zeichen (einschließlich Zeilenumbruch) null Mal oder öfter, gierig.	1	Eingabe: abc def ghi Pattern: %{GREEDYDATA_MULTILINE:data} Ausgabe: {"data": "abc\ndef\nghi"}
ZEICHENFOLGE IN ANFÜHRUNGSGEZEICHEN	Findet Zeichenketten in Anführungszeichen (einfache oder doppelte Anführungszeichen) mit maskierten Zeichen.	20	Eingabe: "Hello, world!" Pattern: %{QUOTEDSTRING:msg} Ausgang: {"msg": "Hello, world!"}
UUID	Entspricht einem Standard-UUID-Format: 8 Hexadezimalzeichen, gefolgt von drei Gruppen von 4 Hexadezimalzeichen und endend mit 12 Hexadezimalzeichen, die alle durch Bindestriche getrennt sind.	20	Eingabe: 550e8400-e29b-41d4-a716-446655440000 Pattern: %{UUID:id} Ausgang: {"id": "550e8400-e29b-41d4-a716-446655440000"}

Grok-Muster	Description	Maximales Musterlimit	Beispiel
URNE	Entspricht der URN-Syntax (Uniform Resource Name).	20	Eingabe: urn:isbn:0451450523 Pattern: %{URN:urn} Ausgang: {"urn": "urn:isbn:0451450523"}

AWS Grok-Muster

Muster	Description	Maximales Musterlimit	Beispiel
ARN	Stimmt mit AWS Amazon-Ressourcennamen (ARNs) überein und erfasst die Partition (aws,aws-cn, oderaws-us-gov), den Service, die Region, die Konto-ID und bis zu 5 hierarchische Ressourcen-IDs, getrennt durch Schrägstriche. Die fehlenden Informationen zwischen Doppelpunkten ARNs werden nicht abgeglichen.	5	Eingabe: arn:aws:iam:us-east-1:123456789012:user/johndoe Pattern: %{ARN:arn} Ausgang: {"arn": "arn:aws:iam:us-east-1:123456789012:user/johndoe"}

Grok-Muster im Netzwerk

Grok-Muster	Description	Maximales Musterlimit	Beispiel
CISCOMAC	Entspricht einer MAC-Adresse im 4-4-4-Hexadezimalformat.	20	Eingabe: 0123.4567.89AB Pattern: <code>%{CISCOMAC:MacAddress}</code> Ausgang: <code>{"MacAddress": "0123.4567.89AB"}</code>
WINDOWS, MAC	Entspricht einer MAC-Adresse im Hexadezimalformat mit Bindestrichen	20	Eingabe: 01-23-45-67-89-AB Pattern: <code>%{WINDOWS MAC:MacAddress}</code> Ausgang: <code>{"MacAddress": "01-23-45-67-89-AB"}</code>
COMMONMAC	Entspricht einer MAC-Adresse im Hexadezimalformat mit Doppelpunkten.	20	Eingabe: 01:23:45:67:89:AB Pattern: <code>%{COMMONMAC:MacAddress}</code> Ausgang: <code>{"MacAddress": "01:23:45:67:89:AB"}</code>
MAC	Entspricht einem der CISCOMAC-, WINDOWSMAC- oder COMMONMAC-Grok-Muster	20	Eingabe: 01:23:45:67:89:AB Pattern: <code>%{MAC:m1}</code> Ausgang: <code>{"m1": "01:23:45:67:89:AB"}</code>
IPV6	Stimmt mit IPv6 Adressen überein, einschließlich komprimierter Formulare und	5	Eingabe: 2001:db8:3333:4444:5555:6666:7777:8888

Grok-Muster	Description	Maximales Musterlimit	Beispiel
	Adressen mit IPv4 -Zuordnung IPv6 .		Pattern: <code>%{IPV6:ip}</code> Ausgang: <code>{"ip": "2001:db8:3333:4444:5555:6666:7777:8888"}</code>
IPv4	Stimmt mit einer IPv4 Adresse überein.	20	Eingabe: <code>192.168.0.1</code> Pattern: <code>%{IPV4:ip}</code> Ausgang: <code>{"ip": "192.168.0.1"}</code>
IP	IPv6 Entspricht entweder Adressen, die von% {IPv6} unterstützt werden, oder IPv4 Adressen, die von% {IPv4} unterstützt werden	5	Eingabe: <code>192.168.0.1</code> Pattern: <code>%{IP:ip}</code> Ausgang: <code>{"ip": "192.168.0.1"}</code>
HOSTNAME oder HOST	Entspricht Domainnamen, einschließlich Subdomänen	5	Eingabe: <code>server-01.internal-network.local</code> Pattern: <code>%{HOST:host}</code> Ausgang: <code>{"host": "server-01.internal-network.local"}</code>

Grok-Muster	Description	Maximales Musterlimit	Beispiel
IP ODER HOST	Entspricht entweder einem Hostnamen oder einer IP-Adresse	5	Eingabe: 2001:db8:3333:4444:5555:6666:7777:8888 Pattern: %{IPORHOST:ip} Ausgang: {"ip": "2001:db8:3333:4444:5555:6666:7777:8888"}
HOSTPORT	Entspricht einer IP-Adresse oder einem Hostnamen, wie sie vom Muster% {IPORHOST} unterstützt werden, gefolgt von einem Doppelpunkt und einer Portnummer, wobei der Port in der Ausgabe als „PORT“ erfasst wird.	5	Eingabe: 192.168.1.1:8080 Pattern: %{HOSTPORT:ip} Ausgang: {"ip": "192.168.1.1:8080", "PORT": "8080"}
URIHOST	Entspricht einer IP-Adresse oder einem Hostnamen, wie sie vom Muster% {IPORHOST} unterstützt werden, optional gefolgt von einem Doppelpunkt und einer Portnummer, wobei der Port als „Port“ erfasst wird, falls vorhanden.	5	Eingabe: example.com:443 10.0.0.1 Pattern: %{URIHOST:host} %{URIHOST:ip} Ausgang: {"host": "example.com:443", "port": "443", "ip": "10.0.0.1"}

Muster für Pfadwechsel

Grok-Muster	Description	Maximales Musterlimit	Beispiel
UNIXPATH	Entspricht URL-Pfaden, möglicherweise einschließlich Abfrageparametern.	20	Eingabe: /search?q=regex Pattern: %{UNIXPATH:path} Ausgang: {"path":"/search?q=regex"}
WINPATH	Entspricht Windows-Dateipfaden.	5	Eingabe: C:\Users\John\Documents\file.txt Pattern: %{WINPATH:path} Ausgang: {"path": "C:\\Users\\John\\Documents\\file.txt"}
PATH	Entspricht entweder URL- oder Windows-Dateipfaden	5	Eingabe: /search?q=regex Pattern: %{PATH:path} Ausgang: {"path":"/search?q=regex"}
TTY	Entspricht Unix-Gerätepfaden für Terminals und Pseudoterminale.	20	Eingabe: /dev/tty1 Pattern: %{TTY:path} Ausgang: {"path":"/dev/tty1"}
URIPROTO	Entspricht Buchstaben, optional gefolgt von einem Pluszeichen (+) und zusätzlic	20	Eingabe: web+transformer

Grok-Muster	Description	Maximales Musterlimit	Beispiel
	hen Buchstaben oder Pluszeichen (+)		Pattern: <code>%{URIPROT0:protocol}</code> Ausgang: <code>{"protocol": "web+transformer"}</code>
URIPATH	Entspricht der Pfadkomponente einer URI	20	Eingabe: <code>/category/sub-category/product_name</code> Pattern: <code>%{URIPATH:path}</code> Ausgang: <code>{"path": "/category/sub-category/product_name"}</code>
URIPARAM	Entspricht den URL-Abfrageparametern	5	Eingabe: <code>?param1=value1&param2=value2</code> Pattern: <code>%{URIPARAM:url}</code> Ausgang: <code>{"url": "?param1=value1&param2=value2"}</code>

Grok-Muster	Description	Maximales Musterlimit	Beispiel
URIPATHPARAM	Entspricht einem URI-Pfad, optional gefolgt von Abfrageparametern	5	Eingabe: /category/sub-category/product?id=12345&color=red Pattern: %{URIPATHPARAM:path} Ausgang: {"path":"/category/sub-category/product?id=12345&color=red"}
URI	Entspricht einer vollständigen URI	5	Eingabe: https://user:password@example.com/path/to/resource?param1=value1&param2=value2 Pattern: %{URI:uri} Ausgang: {"path":"https://user:password@example.com/path/to/resource?param1=value1&param2=value2"}

Grok-Muster für Datum und Uhrzeit

Grok-Muster	Description	Maximales Musterlimit	Beispiel
MONAT	Entspricht vollständigen oder abgekürzten	20	Eingabe: Jan

Grok-Muster	Description	Maximales Musterlimit	Beispiel
	englischen Monatsnamen als ganze Wörter		Pattern: <code>%{MONTH:m onth}</code> Ausgang: <code>{"month": "Jan"}</code> Eingabe: January Pattern: <code>%{MONTH:m onth}</code> Ausgang: <code>{"month": "January"}</code>
MONAT/ ZAHL	Entspricht Monatszahlen von 1 bis 12, mit optionaler führender Null für einstellige Monate.	20	Eingabe: 5 Pattern: <code>%{MONTHNUM:month}</code> Ausgang: <code>{"month": "5"}</code> Eingabe: 05 Pattern: <code>%{MONTHNUM:month}</code> Ausgang: <code>{"month": "05"}</code>
MONTHNUM	Entspricht zweistelligen Monatszahlen von 01 bis 12.	20	Eingabe: 05 Pattern: <code>%{MONTHNUM2:month}</code> Ausgang: <code>{"month": "05"}</code>

Grok-Muster	Description	Maximales Musterlimit	Beispiel
MONAT/ TAG	Entspricht dem Tag des Monats von 1 bis 31, optional mit vorangestellter Null.	20	Eingabe: 31 Pattern: <code>%{MONTHDAY:monthDay}</code> Ausgang: <code>{"monthDay": "31"}</code>
JAHR	Entspricht dem Jahr zweistellig oder vierstellig	20	Eingabe: 2024 Pattern: <code>%{YEAR:year}</code> Ausgang: <code>{"year": "2024"}</code> Eingabe: 24 Pattern: <code>%{YEAR:year}</code> Ausgang: <code>{"year": "24"}</code>
TAG	Entspricht vollständigen oder abgekürzten Tagesnamen.	20	Eingabe: Tuesday Pattern: <code>%{DAY:day}</code> Ausgang: <code>{"day": "Tuesday"}</code>

Grok-Muster	Description	Maximales Musterlimit	Beispiel
STUNDE	Entspricht der Stunde im 24-Stunden-Format mit einer optionalen führenden Null (0) 0—23.	20	Eingabe: 22 Pattern: <code>%{HOUR:hour}</code> Ausgang: <code>{"hour": "22"}</code>
MINUTE	Stimmt mit Minuten überein (00-59).	20	Eingabe: 59 Pattern: <code>%{MINUTE:min}</code> Ausgang: <code>{"min": "59"}</code>

Grok-Muster	Description	Maximales Musterlimit	Beispiel
SECOND	Entspricht einer Zahl, die Sekunden (0) 0-60 darstellt, optional gefolgt von einem Dezimalpunkt oder Doppelpunkt und einer oder mehreren Ziffern für Minutenbruchteile	20	Eingabe: 3 Pattern: <code>%{SECOND:second}</code> Ausgang: <code>{"second": "3"}</code> Eingabe: 30.5 Pattern: <code>%{SECOND:minSec}</code> Ausgang: <code>{"minSec": "30.5"}</code> Eingabe: 30:5 Pattern: <code>%{SECOND:minSec}</code> Ausgang: <code>{"minSec": "30:5"}</code>
TIME	Entspricht einem Zeitformat mit Stunden, Minuten und Sekunden im Format (H) H:mm:(s) s. Sekunden beinhalten Schaltsekunden (0) 0—60.	20	Eingabe: 09:45:32 Pattern: <code>%{TIME:time}</code> Ausgang: <code>{"time": "09:45:32"}</code>

Grok-Muster	Description	Maximales Musterlimit	Beispiel
DATE_US	Entspricht einem Datum im Format (M) M/ (d) d/ (yy) yy oder (M) M- (d) d- (yy) yy.	20	Eingabe: 11/23/2024 Pattern: <code>%{DATE_US:date}</code> Ausgang: <code>{"date": "11/23/2024"}</code> Eingabe: 1-01-24 Pattern: <code>%{DATE_US:date}</code> Ausgang: <code>{"date": "1-01-24"}</code>
DATE_EU	Entspricht dem Datum im Format (d) d/ (M) M/ (yy) yy, (d) d- (M) M- (yy) yy oder (d) d. (M) M. (yy) yy.	20	Eingabe: 23/11/2024 Pattern: <code>%{DATE_EU:date}</code> Ausgang: <code>{"date": "23/11/2024"}</code> Eingabe: 1.01.24 Pattern: <code>%{DATE_EU:date}</code> Ausgang: <code>{"date": "1.01.24"}</code>

Grok-Muster	Description	Maximales Musterlimit	Beispiel
ISO8601_ZEITZONE	Entspricht dem UTC-Offset 'Z' oder dem Zeitzone-Offset mit optionalem Doppelpunkt im Format [+ -] (H) H (:mm).	20	Eingabe: +05:30 Pattern: <code>%{ISO8601_TIMEZONE:tz}</code> Ausgang: <code>{"tz":"+05:30"}</code> Eingabe: -530 Pattern: <code>%{ISO8601_TIMEZONE:tz}</code> Ausgang: <code>{"tz":"-530"}</code> Eingabe: Z Pattern: <code>%{ISO8601_TIMEZONE:tz}</code> Ausgang: <code>{"tz":"Z"}</code>
ISO8601_SEKUNDE	Entspricht einer Zahl, die Sekunden (0) 0-60 darstellt, optional gefolgt von einem Dezimalpunkt oder Doppelpunkt und einer oder mehreren Ziffern für Sekundenbruchteile	20	Eingabe: 60 Pattern: <code>%{ISO8601_SECOND:second}</code> Ausgang: <code>{"second":"60"}</code>

Grok-Muster	Description	Maximales Musterlimit	Beispiel
TIMESTAMP _ISO8601	Entspricht dem ISO8601 Datetime-Format (yy) yy- (M) M- (d) dT (H) H:mm : (s) s (Z [+ -] (H) H:mm) mit optionalen Sekunden und Zeitzone.	20	Eingabe: 2023-05-15T14:30:00+05:30 Pattern: <code>%{TIMESTAMP_ISO8601 1:timestamp}</code> Ausgang: <code>{"timestamp": "2023-05-15T14:30:00+05:30"}</code> Eingabe: 23-5-1T1:25+5:30 Pattern: <code>%{TIMESTAMP_ISO8601 1:timestamp}</code> Ausgang: <code>{"timestamp": "23-5-1T1:25+5:30"}</code> Eingabe: 23-5-1T1:25Z Pattern: <code>%{TIMESTAMP_ISO8601 1:timestamp}</code> Ausgang: <code>{"timestamp": "23-5-1T1:25Z"}</code>

Grok-Muster	Description	Maximales Musterlimit	Beispiel
DATE	Entspricht entweder einem Datum im US-Format mit% {DATE_US} oder einem Datum im EU-Format mit% {DATE_EU}	20	Eingabe: 11/29/2024 Pattern: %{DATE:date} Ausgang: {"date": "11/29/2024"} Eingabe: 29.11.2024 Pattern: %{DATE:date} Ausgang: {"date": "29.11.2024"}
DATUMSSTEMPEL	Entspricht dem Muster % {DATE} gefolgt vom Muster% {TIME}, getrennt durch Leerzeichen oder Bindestrich.	20	Eingabe: 29-11-2024 14:30:00 Pattern: %{DATE:timestamp:dateTime} Ausgang: {"dateTime": "29-11-2024 14:30:00"}
TZ	Entspricht gängigen Abkürzungen für Zeitzone (PST, PDT, MST, MDT, CST CDT, EST, EDT, UTC).	20	Eingabe: PDT Pattern: %{TZ:tz} Ausgang: {"tz": "PDT"}

Grok-Muster	Description	Maximales Musterlimit	Beispiel
DATESTAMP_RFC822	Entspricht Datum und Uhrzeit im folgenden Format: Tag MonthName (D) D (YY) YY (H) H:mm : (s) s Zeitzone	20	Eingabe: Monday Jan 5 23 1:30:00 CDT Pattern: <code>%{DATESTAMP_RFC822:dateTime}</code> Ausgang: <code>{"dateTime": "Monday Jan 5 23 1:30:00 CDT"}</code> Eingabe: Mon January 15 2023 14:30:00 PST Pattern: <code>%{DATESTAMP_RFC822:dateTime}</code> Ausgang: <code>{"dateTime": "Mon January 15 2023 14:30:00 PST"}</code>

Grok-Muster	Description	Maximales Musterlimit	Beispiel
DATESTAMP_RFC2822	Entspricht dem RFC2822 Datums-/Uhrzeitformat: Tag, (d) d MonthName (yy) yy (H) H:mm :(s) s Z [+ -] (H) H:mm	20	Eingabe: Mon, 15 May 2023 14:30:00 +0530 Pattern: %{\DATESTAMP_RFC2822:dateTime} Ausgang: {"dateTime": "Mon, 15 May 2023 14:30:00 +0530"} Eingabe: Monday, 15 Jan 23 14:30:00 Z Pattern: %{\DATESTAMP_RFC2822:dateTime} Ausgang: {"dateTime": "Monday, 15 Jan 23 14:30:00 Z"}
DATESTAMP_OTHER	Entspricht Datum und Uhrzeit im folgenden Format: Tag MonthName (d) d (H) h:mm :(s) s Zeitzone (yy) yy	20	Eingabe: Mon May 15 14:30:00 PST 2023 Pattern: %{\DATESTAMP_OTHER:dateTime} Ausgang: {"dateTime": "Mon May 15 14:30:00 PST 2023"}

Grok-Muster	Description	Maximales Musterlimit	Beispiel
DATESTAMP_EVENTLOG	Entspricht dem kompakten Datetime-Format ohne Trennzeichen: (yy) yymm (d) d (H) Hmm (s) s	20	Eingabe: 20230515143000 Pattern: %{DATESTAMP_EVENTLOG:dateTime} Ausgang: {"dateTime": "20230515143000"}

Loggen Sie die Grok-Muster ein

Grok-Muster	Description	Maximales Musterlimit	Beispiel
PROTOKOLLEBENE	Entspricht den Standardprotokollebenen mit unterschiedlichen Groß- und Kleinschreibung und Abkürzungen, einschließlich der folgenden: Alert/ALERT Trace/TRACE ,Debug/DEBUG ,Notice/NOTICE ,Info/INFO ,Warn/Warning/WARN/WARNING ,Error/ERR/ERROR , Crit/Critical/CRITICAL Fatal/FATAL Severe/SEVERE Emerg/Eme	20	Eingabe: INFO Pattern: %{LOGLEVEL:logLevel} Ausgang: {"logLevel": "INFO"}

Grok-Muster	Description	Maximales Musterlimit	Beispiel
	rgency/EMERG/ EMERGENCY		
HTTP-DATUM	Entspricht dem Datums- und Uhrzeitformat, das häufig in Protokolldateien verwendet wird. Format: (d) MonthName d/ (yy) yy :(H) H:mm :(s) s Zeitzone: Entspricht vollständigen oder abgekürzten englischen Monatsnamen (Beispiel MonthName: „Jan“ oder „January“) Zeitzone: Entspricht dem% {INT} - Grok-Muster	20	Eingabe: 23/Nov/20 24:14:30:00 +0640 Pattern: %{HTTPDATE:date} Ausgang: {"date": "23/Nov/20 24:14:30:00 +0640"}
SYSLOG-ZE ITSTEMPEL	Entspricht dem Datumsformat mit MonthName (d) d (H) H:mm :(s) s MonthName: Entspricht vollständigen oder abgekürzten englischen Monatsnamen (Beispiel: „Jan“ oder „January“)	20	Eingabe: Nov 29 14:30:00 Pattern: %{SYSLOGTIMESTAMP:dateTime} Ausgang: {"dateTime": "Nov 29 14:30:00"}

Grok-Muster	Description	Maximales Musterlimit	Beispiel
PROG	Entspricht einem Programmnamen, der aus einer Zeichenfolge aus Buchstaben, Ziffern, Punkten, Unterstrichen, Schrägstrichen, Prozentzeichen und Bindestrichen besteht.	20	Eingabe: <code>user.profile/settings-page</code> Pattern: <code>%{PROG:program}</code> Ausgang: <code>{"program": "user.profile/settings-page"}</code>
SYSLOGPROG	Entspricht dem PROG-Grok-Muster, optional gefolgt von einer Prozess-ID in eckigen Klammern.	20	Eingabe: <code>user.profile/settings-page[1234]</code> Pattern: <code>%{SYSLOGPROGRAM:programWithId}</code> Ausgang: <code>{"programWithId": "user.profile/settings-page[1234]", "program": "user.profile/settings-page", "pid": "1234"}</code>

Grok-Muster	Description	Maximales Musterlimit	Beispiel
SYSLOGHOST	Entspricht entweder einem% {HOST} - oder% {IP} -Muster	5	Eingabe: 2001:db8:3333:4444:5555:6666:7777:8888 Pattern: %{SYSLOGHOST}:ip} Ausgang: {"ip": "2001:db8:3333:4444:5555:6666:7777:8888"}
SYSLOG-LEVELRICHTUNG	Entspricht der Syslog-Priorität im Dezimalformat. Der Wert sollte in eckige Klammern (<>) eingeschlossen werden.	20	Eingabe: <13.6> Pattern: %{SYSLOGFACILITY}:syslog} Ausgang: {"syslog": "<13.6>", "facility": "13", "priority": "6"}

Gängige Log-Grok-Muster

Sie können vordefinierte benutzerdefinierte Grok-Muster verwenden, um den Protokollformaten von Apache, NGINX und Syslog Protocol (RFC 5424) zu entsprechen. Wenn Sie diese spezifischen Muster verwenden, müssen sie die ersten Muster in Ihrer passenden Konfiguration sein, und ihnen können keine anderen Muster vorausgehen. Außerdem können Sie ihnen nur mit genau einem DATA folgen. GREEDYDATA- oder GREEDYDATA_MULTILINE-Muster.

Grok-Muster	Description	Maximaler Grenzwert für Muster
APACHE_ACCESS_LOG	Entspricht den Apache-Zugriffsprotokollen	1
NGINX_ACCESS_LOG	Entspricht den NGINX-Zugriffsprotokollen	1
SYSLOG5424	Entspricht den Protokollen des Syslog-Protokolls (RFC 5424)	1

Im Folgenden finden Sie gültige und ungültige Beispiele für die Verwendung dieser gängigen Protokollformatmuster.

```

"%{NGINX_ACCESS_LOG} %{DATA}" // Valid
"%{SYSLOG5424}%{DATA:logMsg}" // Valid
"%{APACHE_ACCESS_LOG} %{GREEDYDATA:logMsg}" // Valid
"%{APACHE_ACCESS_LOG} %{SYSLOG5424}" // Invalid (multiple common log patterns used)
"%{NGINX_ACCESS_LOG} %{NUMBER:num}" // Invalid (Only GREEDYDATA and DATA patterns are supported with common log patterns)
"%{GREEDYDATA:logMsg} %{SYSLOG5424}" // Invalid (GREEDYDATA and DATA patterns are supported only after common log patterns)

```

Beispiele für gängige Protokollformate

Beispiel für ein Apache-Protokoll

Beispielprotokoll:

```
127.0.0.1 - - [03/Aug/2023:12:34:56 +0000] "GET /page.html HTTP/1.1" 200 1234
```

Transformator:

```
[
  {
    "grok": {
      "match": "%{APACHE_ACCESS_LOG}"
    }
  }
]
```

Ausgabe:

```
{
  "request": "/page.html",
  "http_method": "GET",
  "status_code": 200,
  "http_version": "1.1",
  "response_size": 1234,
  "remote_host": "127.0.0.1",
  "timestamp": "2023-08-03T12:34:56Z"
}
```

Beispiel für ein NGINX-Protokoll

Beispielprotokoll:

```
192.168.1.100 - Foo [03/Aug/2023:12:34:56 +0000] "GET /account/login.html HTTP/1.1"
200 42 "https://www.amazon.com/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
```

Transformator:

```
[
  {
    "grok": {
      "match": "%{NGINX_ACCESS_LOG}"
    }
  }
]
```

Ausgabe:

```
{
```

```

    "request": "/account/login.html",
    "referrer": "https://www.amazon.com/",
    "agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
    Gecko) Chrome/92.0.4515.131 Safari/537.36",
    "http_method": "GET",
    "status_code": 200,
    "auth_user": "Foo",
    "http_version": "1.1",
    "response_size": 42,
    "remote_host": "192.168.1.100",
    "timestamp": "2023-08-03T12:34:56Z"
  }

```

Beispiel für ein Syslog-Protokoll (RFC 5424)

Beispielprotokoll:

```

<165>1 2003-10-11T22:14:15.003Z mymachine.example.com evntslog - ID47
[exampleSDID@32473 iut="3" eventSource= "Application" eventID="1011"]
[examplePriority@32473 class="high"]

```

Transformator:

```

[
  {
    "grok": {
      "match": "%{SYSLOG5424}"
    }
  }
]

```

Ausgabe:

```

{
  "pri": 165,
  "version": 1,
  "timestamp": "2003-10-11T22:14:15.003Z",
  "hostname": "mymachine.example.com",
  "app": "evntslog",
  "msg_id": "ID47",
  "structured_data": "exampleSDID@32473 iut=\"3\" eventSource= \"Application\" eventID=
  \"1011\"",
  "message": "[examplePriority@32473 class=\"high\"]"
}

```

}

CSV

Der CSV-Prozessor analysiert kommagetrennte Werte (CSV) aus den Protokollereignissen in Spalten.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Quelle	Pfad zu dem Feld im Protokollereignis, das analysiert werden soll	Nein	@message	Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
delimiter	Das Zeichen, das zur Trennung der einzelnen Spalten im ursprünglichen Protokollereignis mit kommagetrennten Werten verwendet wurde	Nein	,	Maximale Länge: 1, es sei denn, der Wert ist oder \t \s
Zeichen in Anführungszeichen	Zeichen, das als Textqualifizierer für eine einzelne Datenspalte verwendet wird	Nein	"	Maximale Länge: 1
Spalten	Liste der Namen, die für die Spalten im transformierten Protokollereignis verwendet werden sollen.	Nein	[column_1, column_2]	Maximale Anzahl von CSV-Spalten: 100 Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
Ziel	Das übergeordnete Feld, unter das transformierte Schlüssel-Wert-Paare gestellt werden sollen	Nein	Root node	Höchstlänge: 128

Feld	Description	Erforderlich?	Standard	Einschränkungen
				Maximale Tiefe verschachtelter Schlüssel: 3

Die Einstellung `delimiter \t` auf trennt jede Spalte durch ein Tabulatorzeichen und `\t` jede Spalte durch ein einzelnes Leerzeichen.

Beispiel

Angenommen, ein Teil eines aufgenommenen Protokollereignisses sieht wie folgt aus:

```
'Akua Mansa':28:'New York: USA'
```

Nehmen wir an, wir verwenden nur den CSV-Prozessor:

```
[
  {
    "csv": {
      "delimiter": ":",
      "quoteCharacter": ""
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "column_1": "Akua Mansa",
  "column_2": "28",
  "column_3": "New York: USA"
}
```

Beispiel 2

Angenommen, ein aufgenommenes Protokollereignis sieht wie folgt aus:

```
{
```

```
"timestamp": "2024-11-23T16:03:12Z",
"type": "user_data",
"logMsg": "'Akua Mansa':28:'New York: USA'"
}
```

Angenommen, wir analysieren das Ereignis als JSON, sie analysieren ein JSON-Feld mit dem CSV-Prozessor und geben Spaltennamen und Ziel an:

```
[
  {
    "parseJSON": {}
  },
  {
    "csv": {
      "source": "logMsg",
      "delimiter": ":",
      "quoteCharacter": "'",
      "columns": ["name", "age", "location"],
      "destination": "msg"
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "timestamp": "2024-11-23T16:03:12Z",
  "logMsg": "'Akua Mansa':28:'New York: USA'",
  "type": "user_data",
  "msg": {
    "name": "Akua Mansa",
    "age": "28",
    "location": "New York: USA"
  }
}
```

parseKeyValue

Verwenden Sie den `parseKeyValue` Prozessor, um ein bestimmtes Feld in Schlüssel-Wert-Paare zu zerlegen. Sie können den Prozessor mit den folgenden Optionen so anpassen, dass er Feldinformationen analysiert.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Quelle	Pfad zu dem Feld im Protokollereignis, das analysiert werden soll	Nein	@message	Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
Ziel	Das Zielfeld, in das die extrahierten Schlüssel-Wert-Paare eingefügt werden sollen	Nein		Höchstlänge: 128
FieldDelimiter	Die Feldtrennzeichenfolge, die zwischen Schlüssel-Wert-Paaren in den ursprünglichen Protokollereignissen verwendet wird	Nein	&	Höchstlänge: 128
keyValueDelimiter	Die Trennzeichenfolge, die zwischen dem Schlüssel und dem Wert in jedem Paar im transformierten Protokollereignis verwendet werden soll	Nein	=	Höchstlänge: 128
nonMatchValue	Ein Wert, der in das Wertfeld im Ergebnis eingefügt werden soll, wenn ein Schlüssel-Wert-Paar nicht erfolgreich aufgeteilt wurde.	Nein		Höchstlänge: 128
keyPrefix	Wenn Sie allen transformierten Schlüsseln ein Präfix hinzufügen möchten, geben Sie es hier an.	Nein		Höchstlänge: 128
overwriteIfExists	Ob der Wert überschrieben werden soll, wenn der Zielschlüssel bereits existiert	Nein	false	

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
key1:value1!key2:value2!key3:value3!key4
```

Angenommen, wir verwenden die folgende Prozessorkonfiguration:

```
[
  {
    "parseKeyValue": {
      "destination": "new_key",
      "fieldDelimiter": "!",
      "keyValueDelimiter": ":",
      "nonMatchValue": "defaultValue",
      "keyPrefix": "parsed_"
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "new_key": {
    "parsed_key1": "value1",
    "parsed_key2": "value2",
    "parsed_key3": "value3",
    "parsed_key4": "defaultValue"
  }
}
```

Integrierte Prozessoren für AWS verkaufte Logs

Dieser Abschnitt enthält Informationen zu den integrierten Prozessoren, die Sie mit AWS Diensten verwenden können, die Protokolle versenden.

Inhalt

- [Analysieren Sie WAF](#)
- [Analysieren Sie PostGres](#)
- [Analysieren Sie CloudFront](#)
- [ParseRoute53](#)

- [ParseVPC](#)
- [Parse nach OCSF](#)

Analysieren Sie WAF

Verwenden Sie diesen Prozessor, um verkaufte Logs zu analysieren AWS WAF . Er nimmt den Inhalt von jedem Header-Namen `httpRequest.headers` und erstellt daraus JSON-Schlüssel mit dem entsprechenden Wert. Das Gleiche tut er auch für `labels`. Diese Transformationen können das Abfragen von AWS WAF Protokollen erheblich vereinfachen. Weitere Informationen zum AWS WAF Protokollformat finden Sie unter [Protokollbeispiele für Web-ACL-Verkehr](#).

Dieser Prozessor akzeptiert nur @message die Eingabe.

Important

Wenn Sie diesen Prozessor verwenden, muss es der erste Prozessor in Ihrem Transformator sein.

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
  "timestamp": 1576280412771,
  "formatVersion": 1,
  "webaclId": "arn:aws:wafv2:ap-southeast-2:111122223333:regional/webacl/
STMTest/1EXAMPLE-2ARN-3ARN-4ARN-123456EXAMPLE",
  "terminatingRuleId": "STMTest_SQLi_XSS",
  "terminatingRuleType": "REGULAR",
  "action": "BLOCK",
  "terminatingRuleMatchDetails": [
    {
      "conditionType": "SQL_INJECTION",
      "sensitivityLevel": "HIGH",
      "location": "HEADER",
      "matchedData": ["10", "AND", "1"]
    }
  ],
  "httpSourceName": "-",
```

```
"httpSourceId": "-",
"ruleGroupList": [],
"rateBasedRuleList": [],
"nonTerminatingMatchingRules": [],
"httpRequest": {
  "clientId": "1.1.1.1",
  "country": "AU",
  "headers": [
    { "name": "Host", "value": "localhost:1989" },
    { "name": "User-Agent", "value": "curl/7.61.1" },
    { "name": "Accept", "value": "*/*" },
    { "name": "x-stm-test", "value": "10 AND 1=1" }
  ],
  "uri": "/myUri",
  "args": "",
  "httpVersion": "HTTP/1.1",
  "httpMethod": "GET",
  "requestId": "rid"
},
"labels": [{ "name": "value" }]
}
```

Die Prozessorkonfiguration sieht wie folgt aus:

```
[
  {
    "parseWAF": {}
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "httpRequest": {
    "headers": {
      "Host": "localhost:1989",
      "User-Agent": "curl/7.61.1",
      "Accept": "*/*",
      "x-stm-test": "10 AND 1=1"
    },
    "clientId": "1.1.1.1",
    "country": "AU",
    "uri": "/myUri",
```

```
    "args": "",
    "httpVersion": "HTTP/1.1",
    "httpMethod": "GET",
    "requestId": "rid"
  },
  "labels": { "name": "value" },
  "timestamp": 1576280412771,
  "formatVersion": 1,
  "webaclId": "arn:aws:wafv2:ap-southeast-2:111122223333:regional/webacl/
STMTest/1EXAMPLE-2ARN-3ARN-4ARN-123456EXAMPLE",
  "terminatingRuleId": "STMTest_SQLi_XSS",
  "terminatingRuleType": "REGULAR",
  "action": "BLOCK",
  "terminatingRuleMatchDetails": [
    {
      "conditionType": "SQL_INJECTION",
      "sensitivityLevel": "HIGH",
      "location": "HEADER",
      "matchedData": ["10", "AND", "1"]
    }
  ],
  "httpSourceName": "-",
  "httpSourceId": "-",
  "ruleGroupList": [],
  "rateBasedRuleList": [],
  "nonTerminatingMatchingRules": []
}
```

Analysieren Sie PostGres

Verwenden Sie diesen Prozessor, um verkaufte Protokolle zu analysieren Amazon RDS for PostgreSQL, Felder zu extrahieren und sie in das JSON-Format zu konvertieren. Weitere Informationen zum Protokollformat RDS für PostgreSQL finden Sie unter Datenbankprotokolldateien von [RDS for PostgreSQL](#).

Dieser Prozessor akzeptiert nur @message als Eingabe.

Important

Wenn Sie diesen Prozessor verwenden, muss es der erste Prozessor in Ihrem Transformator sein.

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
2019-03-10 03:54:59 UTC:10.0.0.123(52834):postgres@logtestdb:[20175]:ERROR: column
"wrong_column_name" does not exist at character 8
```

Die Prozessorkonfiguration sieht wie folgt aus:

```
[
  {
    "parsePostgres": {}
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "logTime": "2019-03-10 03:54:59 UTC",
  "srcIp": "10.0.0.123(52834)",
  "userName": "postgres",
  "dbName": "logtestdb",
  "processId": "20175",
  "logLevel": "ERROR"
}
```

Analysieren Sie CloudFront

Verwenden Sie diesen Prozessor, um verkaufte Protokolle zu analysieren, Felder Amazon CloudFront zu extrahieren und sie in das JSON-Format zu konvertieren. Kodierte Feldwerte werden dekodiert. Werte, bei denen es sich um Ganzzahlen und Doppelzahlen handelt, werden als solche behandelt. Weitere Informationen zum Amazon CloudFront Protokollformat finden [Sie unter Standardprotokolle \(Zugriffsprotokolle\) konfigurieren und verwenden](#).

Dieser Prozessor akzeptiert nur @message als Eingabe.

Important

Wenn Sie diesen Prozessor verwenden, muss es der erste Prozessor in Ihrem Transformator sein.

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
2019-12-04 21:02:31 LAX1 392 192.0.2.24 GET
d111111abcdef8.cloudfront.net /index.html 200 - Mozilla/5.0%20(Windows
%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,
%20like%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Hit
S0X4xwn4XV6Q4rgb7XiVG0Hms_BG1TAC4KyHmureZmBNrjGdRLiNIQ==
d111111abcdef8.cloudfront.net https 23 0.001 - TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256 Hit HTTP/2.0 - - 11040 0.001 Hit text/html 78 - -
```

Die Prozessorkonfiguration sieht wie folgt aus:

```
[
  {
    "parseCloudfront": {}
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "date": "2019-12-04",
  "time": "21:02:31",
  "x-edge-location": "LAX1",
  "sc-bytes": 392,
  "c-ip": "192.0.2.24",
  "cs-method": "GET",
  "cs(Host)": "d111111abcdef8.cloudfront.net",
  "cs-uri-stem": "/index.html",
  "sc-status": 200,
  "cs(Referer)": "-",
  "cs(User-Agent)": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36",
  "cs-uri-query": "-",
  "cs(Cookie)": "-",
  "x-edge-result-type": "Hit",
  "x-edge-request-id": "S0X4xwn4XV6Q4rgb7XiVG0Hms_BG1TAC4KyHmureZmBNrjGdRLiNIQ==",
  "x-host-header": "d111111abcdef8.cloudfront.net",
  "cs-protocol": "https",
  "cs-bytes": 23,
  "time-taken": 0.001,
```

```
{
  "x-forwarded-for": "-",
  "ssl-protocol": "TLSv1.2",
  "ssl-cipher": "ECDHE-RSA-AES128-GCM-SHA256",
  "x-edge-response-result-type": "Hit",
  "cs-protocol-version": "HTTP/2.0",
  "fle-status": "-",
  "fle-encrypted-fields": "-",
  "c-port": 11040,
  "time-to-first-byte": 0.001,
  "x-edge-detailed-result-type": "Hit",
  "sc-content-type": "text/html",
  "sc-content-len": 78,
  "sc-range-start": "-",
  "sc-range-end": "-"
}
```

ParseRoute53

Verwenden Sie diesen Prozessor, um verkaufte Protokolle zu analysieren Amazon Route 53 Public Data Plane , Felder zu extrahieren und sie in das JSON-Format zu konvertieren. Kodierte Feldwerte werden dekodiert. Dieser Prozessor unterstützt Amazon Route 53 Resolver keine Protokolle.

Dieser Prozessor akzeptiert nur @message als Eingabe.

Important

Wenn Sie diesen Prozessor verwenden, muss es der erste Prozessor in Ihrem Transformator sein.

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
1.0 2017-12-13T08:15:50.235Z Z123412341234 example.com AAAA NOERROR TCP IAD12 192.0.2.0
198.51.100.0/24
```

Die Prozessorkonfiguration sieht wie folgt aus:

```
[
  {
    "parseRoute53": {}
  }
]
```

```
}  
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{  
  "version": 1.0,  
  "queryTimestamp": "2017-12-13T08:15:50.235Z",  
  "hostZoneId": "Z123412341234",  
  "queryName": "example.com",  
  "queryType": "AAAA",  
  "responseCode": "NOERROR",  
  "protocol": "TCP",  
  "edgeLocation": "IAD12",  
  "resolverIp": "192.0.2.0",  
  "ednsClientSubnet": "198.51.100.0/24"  
}
```

ParseVPC

Verwenden Sie diesen Prozessor, um von Amazon VPC verkaufte Protokolle zu analysieren, Felder zu extrahieren und sie in das JSON-Format zu konvertieren. Kodierte Feldwerte werden dekodiert.

Dieser Prozessor akzeptiert nur @message als Eingabe.

Important

Wenn Sie diesen Prozessor verwenden, muss es der erste Prozessor in Ihrem Transformator sein.

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
2 123456789010 eni-abc123de 192.0.2.0 192.0.2.24 20641 22 6 20 4249 1418530010  
1418530070 ACCEPT OK
```

Die Prozessorkonfiguration sieht wie folgt aus:

```
[
```

```
{
  "parseVPC": {}
}
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "version": 2,
  "accountId": "123456789010",
  "interfaceId": "eni-abc123de",
  "srcAddr": "192.0.2.0",
  "dstAddr": "192.0.2.24",
  "srcPort": 20641,
  "dstPort": 22,
  "protocol": 6,
  "packets": 20,
  "bytes": 4249,
  "start": 1418530010,
  "end": 1418530070,
  "action": "ACCEPT",
  "logStatus": "OK"
}
```

Parse nach OCSF

Der `parseToOCSF` Prozessor wandelt Protokolle in Open Cybersecurity Schema Framework (OCSF) -Ereignisse um. OCSF ist ein offener Standard, der ein gemeinsames Schema für Sicherheitsdaten bereitstellt und so eine bessere Interoperabilität und Analyse zwischen verschiedenen Sicherheitstools und Plattformen ermöglicht.

Dieser Prozessor ist besonders nützlich für Sicherheitsanalyse-Workflows, bei denen Sie Protokollformate verschiedener AWS Dienste in einem konsistenten Schema für nachgelagerte Analysen standardisieren müssen.

Parameter

`eventSource` (Erforderlich)

Gibt den AWS Dienst oder Prozess an, der die zu konvertierenden Protokollereignisse erzeugt. Folgende sind gültige Werte:

- `CloudTrail`- CloudTrail Protokolle

- Route53Resolver- Route 53 Resolver-Protokolle
- VPCFlow- Amazon VPC-Flussprotokolle
- EKSAudit- Amazon EKS-Auditprotokolle
- AWSWAF- AWS WAF Protokolle

ocsfVersion (Erforderlich)

Gibt an, welche Version des OCSF-Schemas für die transformierten Protokollereignisse verwendet werden soll. Derzeit unterstützte Versionen: V1.1, V1.5

mappingVersion (optional)

Gibt die Version der OCSF-Transformationszuweisung an. Steuert, welche Transformationslogik bei der Konvertierung von Protokollen in das OCSF-Format angewendet wird. Wenn nicht angegeben, wird die neueste verfügbare Version zum Zeitpunkt der Richtlinienerstellung verwendet. Bestehende Richtlinien werden nicht automatisch aktualisiert, wenn neue Mapping-Versionen veröffentlicht werden. Aktuelle neueste Version: v1.5.0.

Hinweis: Wird nicht unterstützt, wenn ocsfVersion es ist V1.1.

source (optional)

Der Pfad zu dem Feld im Protokollereignis, das Sie analysieren möchten. Wenn es weggelassen wird, wird die gesamte Protokollnachricht analysiert.

Beispiel

Das folgende Beispiel zeigt, wie VPC Flow Logs in das OCSF-Format konvertiert werden:

parseToOCSF

```
{
  "parseToOCSF": {
    "eventSource": "VPCFlow",
    "ocsfVersion": "V1.1"
  }
}
```

Das folgende Beispiel zeigt, wie eine bestimmte Mapping-Version für ein konsistentes Transformationsverhalten angegeben wird:

```
{
  "parseToOCSF": {
```

```
"eventSource": "CloudTrail",
"ocsfVersion": "V1.5",
"mappingVersion": "v1.5.0"
}
}
```

Prozessoren mutieren durch Zeichenketten

Dieser Abschnitt enthält Informationen zu den Prozessoren für Zeichenkettenmutationen, die Sie mit einem Log-Event-Transformator verwenden können.

Inhalt

- [lowerCaseString](#)
- [upperCaseString](#)
- [SplitString](#)
- [SubstituteString](#)
- [TrimString](#)

lowerCaseString

Der lowerCaseString Prozessor konvertiert eine Zeichenfolge in ihre kleingeschriebene Version.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Mit Schlüssel	Eine Liste von Schlüsseln, die in Kleinbuchstaben umgewandelt werden sollen	Ja		Maximale Anzahl von Einträgen: 10

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
  "outer_key": {
    "inner_key": "INNER_VALUE"
  }
}
```

```
}
```

Die Transformer-Konfiguration sieht wie folgt aus und verwendet `lowerCaseString` mit `parseJSON`:

```
[
  {
    "parseJSON": {}
  },
  {
    "lowerCaseString": {
      "withKeys": ["outer_key.inner_key"]
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

upperCaseString

Der `upperCaseString` Prozessor konvertiert eine Zeichenfolge in ihre Großbuchstabenversion.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Mit Schlüssel	Eine Liste von Schlüsseln, die in Großbuchstaben umgewandelt werden sollen	Ja		Maximale Anzahl von Einträgen: 10

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
```

```
"outer_key": {
  "inner_key": "inner_value"
}
```

Die Transformer-Konfiguration sieht wie folgt aus und verwendet `upperCaseString` mit `parseJSON`:

```
[
  {
    "parseJSON": {}
  },
  {
    "upperCaseString": {
      "withKeys": ["outer_key.inner_key"]
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "outer_key": {
    "inner_key": "INNER_VALUE"
  }
}
```

SplitString

Der `splitString` Prozessor ist eine Art von String-Mutate-Prozessor, der ein Feld mithilfe eines Trennzeichens in ein Array aufteilt.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Einträge	Array von Einträgen. Jedes Element im Array muss <code>delimiter</code> Felder enthalten <code>source</code> .	Ja		Maximale Anzahl von Einträgen: 10

Feld	Description	Erforderlich?	Standard	Einschränkungen
Quelle	Der Schlüssel des Feldwerts, der geteilt werden soll	Ja		Höchstlänge: 128
delimiter	Die Trennzeichenfolge, nach der der Feldwert aufgeteilt werden soll	Ja		Höchstlänge: 128

Beispiel 1

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

Die Transformer-Konfiguration sieht wie folgt aus und verwendet `splitString` mit `parseJSON`:

```
[
  {
    "parseJSON": {}
  },
  {
    "splitString": {
      "entries": [
        {
          "source": "outer_key.inner_key",
          "delimiter": "_"
        }
      ]
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
```

```
"outer_key": {
  "inner_key": [
    "inner",
    "value"
  ]
}
```

Beispiel 2

Das Trennzeichen, anhand dessen die Zeichenfolge aufgeteilt werden soll, kann mehrere Zeichen lang sein.

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
  "outer_key": {
    "inner_key": "item1, item2, item3"
  }
}
```

Die Konfiguration des Transformators sieht wie folgt aus:

```
[
  {
    "parseJSON": {}
  },
  {
    "splitString": {
      "entries": [
        {
          "source": "outer_key.inner_key",
          "delimiter": ", "
        }
      ]
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
```

```

"outer_key": {
  "inner_key": [
    "item1",
    "item2",
    "item3"
  ]
}
}

```

SubstituteString

Der `substituteString` Prozessor ist ein Typ eines String-Mutationsprozessors, der den Wert eines Schlüssels mit einem regulären Ausdruck vergleicht und alle Treffer durch eine Ersatzzeichenfolge ersetzt.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Einträge	Reihe von Einträgen. Jedes Element im Array muss to Felder <code>sourcefrom</code> , und enthalten.	Ja		Maximale Anzahl von Einträgen: 10
Quelle	Der Schlüssel des Feldes, das geändert werden soll	Ja		Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
from	Die Zeichenfolge für reguläre Ausdrücke, die ersetzt werden soll. Spezielle reguläre Zeichen wie <code>[und]</code> müssen mit <code>\</code> maskiert werden, wenn doppelte Anführungszeichen verwendet werden, und mit <code>\</code> , wenn einfache Anführungszeichen verwendet werden oder wenn sie über den konfiguriert wurden. AWS-Managementkonsole Weitere Informationen finden	Ja		Höchstlänge: 128

Feld	Description	Erforderlich?	Standard	Einschränkungen
	Sie unter Class Pattern auf der Oracle-Website. Sie können ein Muster einbinden (. . .), um eine nummerierte Erfassungsgruppe und (? P<group_name> . . .) benannte Erfassungsgruppen zu erstellen, auf die im to Feld verwiesen werden kann.			
to	Die Zeichenfolge, die für jede Übereinstimmung von from Rückverweisen auf Erfassungsgruppen ersetzt werden soll, kann verwendet werden. Verwenden Sie die Form \$n für nummerierte Gruppen wie \$1 und \${group_name} für benannte Gruppen wie \$. {my_group} >	Ja		Höchstlänge: 128 Maximale Anzahl von Rückverweisen: 10 Maximale Anzahl doppelter Rückverweise: 2

Beispiel 1

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
  "outer_key": {
    "inner_key1": "[]",
    "inner_key2": "123-345-567",
    "inner_key3": "A cat takes a catnap."
  }
}
```

Die Transformer-Konfiguration sieht wie folgt aus und verwendet `substituteString` mit `parseJSON`:

```
[
  {
    "parseJSON": {}
  },
  {
    "substituteString": {
      "entries": [
        {
          "source": "outer_key.inner_key1",
          "from": "\\[\\]",
          "to": "value1"
        },
        {
          "source": "outer_key.inner_key2",
          "from": "[0-9]{3}-[0-9]{3}-[0-9]{3}",
          "to": "xxx-xxx-xxx"
        },
        {
          "source": "outer_key.inner_key3",
          "from": "cat",
          "to": "dog"
        }
      ]
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "outer_key": {
    "inner_key1": "value1",
    "inner_key2": "xxx-xxx-xxx",
    "inner_key3": "A dog takes a dognap."
  }
}
```

Beispiel 2

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
```

```

    "outer_key": {
      "inner_key1": "Tom, Dick, and Harry",
      "inner_key2": "arn:aws:sts::123456789012:assumed-role/MyImportantRole/MySession"
    }
  }
}

```

Die Transformer-Konfiguration sieht wie folgt aus und verwendet `substituteString` mit `parseJSON`:

```

[
  {
    "parseJSON": {}
  },
  {
    "substituteString": {
      "entries": [
        {
          "source": "outer_key.inner_key1",
          "from": "(\\w+), (\\w+), and (\\w+)",
          "to": "$1 and $3"
        },
        {
          "source": "outer_key.inner_key2",
          "from": "^arn:aws:sts::(?P<account_id>\\d{12}):assumed-role/(?P<role_name>[\\w+=,\\.@-]+)/(?P<role_session_name>[\\w+=,\\.@-]+)$",
          "to": "${account_id}:${role_name}:${role_session_name}"
        }
      ]
    }
  }
]

```

Das transformierte Protokollereignis wäre das Folgende.

```

{
  "outer_key": {
    "inner_key1": "Tom and Harry",
    "inner_key2": "123456789012:MyImportantRole:MySession"
  }
}

```

TrimString

Der trimString Prozessor entfernt Leerzeichen am Anfang und Ende eines Schlüssels.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Mit Schlüssel	Eine Liste von Schlüsseln zum Trimmen	Ja		Maximale Anzahl an Einträgen: 10

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
  "outer_key": {
    "inner_key": "  inner_value  "
  }
}
```

Die Transformer-Konfiguration sieht wie folgt aus und verwendet trimString mit parseJSON:

```
[
  {
    "parseJSON": {}
  },
  {
    "trimString": {
      "withKeys": ["outer_key.inner_key"]
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

}

JSON-Prozessoren mutieren

Dieser Abschnitt enthält Informationen zu den mutierten JSON-Prozessoren, die Sie mit einem Log-Event-Transformer verwenden können.

Inhalt

- [Schlüssel hinzufügen](#)
- [Schlüssel löschen](#)
- [MoveKeys](#)
- [Schlüssel umbenennen](#)
- [Wert kopieren](#)
- [listToMap](#)

Schlüssel hinzufügen

Verwenden Sie den addKeys Prozessor, um dem Protokollereignis neue Schlüssel-Wert-Paare hinzuzufügen.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Einträge	Reihe von Einträgen. Jedes Element im Array kann <code>overwriteIfExists</code> Felder <code>keyvalue</code> , und enthalten.	Ja		Maximale Anzahl von Einträgen: 5
Schlüssel	Der Schlüssel des neuen Eintrags, der hinzugefügt werden soll	Ja		Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
value	Der Wert des neuen Eintrags, der hinzugefügt werden soll	Ja		Maximale Länge: 256

Feld	Description	Erforderlich?	Standard	Einschränkungen
overwriteIfExists	Wenn Sie dies auf <code>true</code> setzen, wird der bestehende Wert überschrieben, falls er im Ereignis <code>key</code> bereits vorhanden ist. Der Standardwert ist <code>false</code> .	Nein	<code>false</code>	Kein Limit

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

Die Transformer-Konfiguration sieht wie folgt aus und verwendet `addKeys` mit `parseJSON`:

```
[
  {
    "parseJSON": {}
  },
  {
    "addKeys": {
      "entries": [
        {
          "key": "outer_key.new_key",
          "value": "new_value"
        }
      ]
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
```

```
"outer_key": {
  "inner_key": "inner_value",
  "new_key": "new_value"
}
```

Schlüssel löschen

Verwenden Sie den `deleteKeys` Prozessor, um Felder aus einem Protokollereignis zu löschen. Diese Felder können Schlüssel-Wert-Paare enthalten.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Mit Schlüssel	Die Liste der zu löschenden Schlüssel.	Ja	Kein Limit	Maximale Anzahl an Einträgen: 5

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

Die Transformer-Konfiguration sieht wie folgt aus und verwendet `deleteKeys` mit `parseJSON`:

```
[
  {
    "parseJSON": {}
  },
  {
    "deleteKeys": {
      "withKeys": ["outer_key.inner_key"]
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "outer_key": {}
}
```

MoveKeys

Verwenden Sie den moveKeys Prozessor, um eine Taste von einem Feld in ein anderes zu verschieben.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Einträge	Reihe von Einträgen. Jedes Element im Array kann <code>overwriteIfExists</code> Felder <code>sourcetaget</code> , und enthalten.	Ja		Maximale Anzahl von Einträgen: 5
Quelle	Der Schlüssel zum Bewegen	Ja		Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
Ziel	Der Schlüssel, zu dem gewechselt werden soll	Ja		Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
<code>overwriteIfExists</code>	Wenn Sie dies auf <code>setzenttrue</code> , wird der vorhandene Wert überschrieben, falls er im key Ereignis bereits vorhanden ist. Der Standardwert ist <code>false</code> .	Nein	<code>false</code>	Kein Limit

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
  "outer_key1": {
    "inner_key1": "inner_value1"
  },
  "outer_key2": {
    "inner_key2": "inner_value2"
  }
}
```

Die Transformer-Konfiguration sieht wie folgt aus und verwendet `moveKeys` mit `parseJSON`:

```
[
  {
    "parseJSON": {}
  },
  {
    "moveKeys": {
      "entries": [
        {
          "source": "outer_key1.inner_key1",
          "target": "outer_key2"
        }
      ]
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "outer_key1": {},
  "outer_key2": {
    "inner_key2": "inner_value2",
    "inner_key1": "inner_value1"
  }
}
```

Schlüssel umbenennen

Verwenden Sie den `renameKeys` Prozessor, um Schlüssel in einem Protokollereignis umzubenennen.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Einträge	Reihe von Einträgen. Jedes Element im Array kann <code>overwriteIfExists</code> Felder <code>keytarget</code> , und enthalten.	Ja	Kein Limit	Maximale Anzahl von Einträgen: 5
Schlüssel	Der Schlüssel zum Umbenennen	Ja	Kein Limit	Höchstlänge: 128
Ziel	Der neue Schlüsselname	Ja	Kein Limit	Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
<code>overwriteIfExists</code>	Wenn Sie dies auf <code>true</code> setzen, wird der vorhandene Wert überschrieben, falls er im <code>key</code> Ereignis bereits vorhanden ist. Der Standardwert ist <code>false</code> .	Nein	<code>false</code>	Kein Limit

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

Die Transformer-Konfiguration sieht wie folgt aus und verwendet `renameKeys` mit `parseJSON`:

```
[
  {
    "parseJSON": {}
  },
  {
    "renameKeys": {
      "entries": [
        {
          "key": "outer_key",
          "target": "new_key"
        }
      ]
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "new_key": {
    "inner_key": "inner_value"
  }
}
```

Wert kopieren

Verwenden Sie den `copyValue` Prozessor, um Werte innerhalb eines Protokollereignisses zu kopieren. Sie können diesen Prozessor auch verwenden, um Metadaten zu Protokollereignissen hinzuzufügen, indem Sie die Werte der folgenden Metadatenschlüssel in die Protokollereignisse kopieren: `@logGroupName`, `@logGroupStream`, `@accountId`, `@regionName`. Dies wird im folgenden Beispiel veranschaulicht.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Einträge	Reihe von Einträgen. Jedes Element im Array kann <code>overwriteIfExists</code> Felder <code>source</code> <code>target</code> , und enthalten.	Ja		Maximale Anzahl von Einträgen: 5

Feld	Description	Erforderlich?	Standard	Einschränkungen
Quelle	Der Schlüssel zum Kopieren	Ja		Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
Ziel	Der Schlüssel, in den der Wert kopiert werden soll	Ja	Kein Limit	Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
overwriteIfExists	Wenn Sie dies auf <code>true</code> setzen, wird der vorhandene Wert überschrieben, falls er im key Ereignis bereits vorhanden ist. Der Standardwert ist <code>false</code> .	Nein	<code>false</code>	Kein Limit

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

Die Transformer-Konfiguration sieht wie folgt aus und verwendet `copyValue` mit `parseJSON`:

```
[
  {
    "parseJSON": {}
  },
  {
    "copyValue": {
      "entries": [
```

```

        {
            "key": "outer_key.new_key",
            "target": "new_key"
        },
        {
            "source": "@logGroupName",
            "target": "log_group_name"
        },
        {
            "source": "@logGroupStream",
            "target": "log_group_stream"
        },
        {
            "source": "@accountId",
            "target": "account_id"
        },
        {
            "source": "@regionName",
            "target": "region_name"
        }
    ]
}
]

```

Das transformierte Protokollereignis wäre das Folgende.

```

{
  "outer_key": {
    "inner_key": "inner_value"
  },
  "new_key": "inner_value",
  "log_group_name": "myLogGroupName",
  "log_group_stream": "myLogStreamName",
  "account_id": "012345678912",
  "region_name": "us-east-1"
}

```

listToMap

Der `listToMap` Prozessor nimmt eine Liste von Objekten, die Schlüsselfelder enthalten, und konvertiert sie in eine Zuordnung von Zielschlüsseln.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Quelle	Der Schlüssel in der ProcessingEvent mit einer Liste von Objekten, die in eine Map konvertiert werden	Ja		Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
Schlüssel	Der Schlüssel der Felder, die als Schlüssel in der generierten Map extrahiert werden sollen	Ja		Höchstlänge: 128
WertSchlüssel	Wenn dies angegeben ist, werden die Werte, die Sie in diesem Parameter angeben, aus den source Objekten extrahiert und in die Werte der generierten Map eingefügt. Andernfalls werden die Originalobjekte in der Quellliste in die Werte der generierten Map aufgenommen.	Nein		Höchstlänge: 128
Ziel	Der Schlüssel des Feldes, das die generierte Map enthalten soll	Nein	Stammknoten	Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
flatten	Ein boolescher Wert, der angibt, ob die Liste in einzelne Elemente reduziert wird oder ob es sich bei den Werten in der generierten Map um Listen handelt. Standardmäßig werden die Werte für die passenden Schlüssel in einem Array dargestellt. Auf	Nein	false	

Feld	Description	Erforderlich?	Standard	Einschränkungen
	einstellen <code>true</code> , flatten um das Array basierend auf dem Wert von in einen einzelnen Wert umzuwandeln <code>flattenedElement</code> .			
flattenedElement	Wenn Sie flatten auf <code>true</code> einstellen, geben Sie hier <code>flattenedElement</code> an, welches Element <code>first</code> oder <code>last</code> beibehalten werden soll.	Erforderlich, wenn auf <code>gesetzt flatten</code> ist <code>true</code>		Der Wert kann nur <code>first</code> oder sein <code>last</code>

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
  "outer_key": [
    {
      "inner_key": "a",
      "inner_value": "val-a"
    },
    {
      "inner_key": "b",
      "inner_value": "val-b1"
    },
    {
      "inner_key": "b",
      "inner_value": "val-b2"
    },
    {
      "inner_key": "c",
      "inner_value": "val-c"
    }
  ]
}
```

Transformator für Anwendungsfall 1: flatten ist false

```
[
  {
    "parseJSON": {}
  },
  {
    "listToMap": {
      "source": "outer_key"
      "key": "inner_key",
      "valueKey": "inner_value",
      "flatten": false
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "outer_key": [
    {
      "inner_key": "a",
      "inner_value": "val-a"
    },
    {
      "inner_key": "b",
      "inner_value": "val-b1"
    },
    {
      "inner_key": "b",
      "inner_value": "val-b2"
    },
    {
      "inner_key": "c",
      "inner_value": "val-c"
    }
  ],
  "a": [
    "val-a"
  ],
  "b": [
    "val-b1",
    "val-b2"
  ]
}
```

```
    ],  
    "c": [  
        "val-c"  
    ]  
}
```

Transformator für Anwendungsfall 2: `flatten` ist `true` und `flattenedElement` ist `first`

```
[  
  {  
    "parseJSON": {}  
  },  
  {  
    "listToMap": {  
      "source": "outer_key"  
      "key": "inner_key",  
      "valueKey": "inner_value",  
      "flatten": true,  
      "flattenedElement": "first"  
    }  
  }  
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{  
  "outer_key": [  
    {  
      "inner_key": "a",  
      "inner_value": "val-a"  
    },  
    {  
      "inner_key": "b",  
      "inner_value": "val-b1"  
    },  
    {  
      "inner_key": "b",  
      "inner_value": "val-b2"  
    },  
    {  
      "inner_key": "c",  
      "inner_value": "val-c"  
    }  
  ]  
}
```

```
],  
  "a": "val-a",  
  "b": "val-b1",  
  "c": "val-c"  
}
```

Transformator für Anwendungsfall 3: `flatten` ist `true` und `flattenedElement` ist `last`

```
[  
  {  
    "parseJSON": {}  
  },  
  {  
    "listToMap": {  
      "source": "outer_key"  
      "key": "inner_key",  
      "valueKey": "inner_value",  
      "flatten": true,  
      "flattenedElement": "last"  
    }  
  }  
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{  
  "outer_key": [  
    {  
      "inner_key": "a",  
      "inner_value": "val-a"  
    },  
    {  
      "inner_key": "b",  
      "inner_value": "val-b1"  
    },  
    {  
      "inner_key": "b",  
      "inner_value": "val-b2"  
    },  
    {  
      "inner_key": "c",  
      "inner_value": "val-c"  
    }  
  ]  
}
```

```
],  
  "a": "val-a",  
  "b": "val-b2",  
  "c": "val-c"  
}
```

Prozessoren für Datentypkonverter

Dieser Abschnitt enthält Informationen zu den Datentypkonverter-Prozessoren, die Sie mit einem Log-Event-Transformator verwenden können.

Inhalt

- [TypeConverter](#)
- [DateTimeConverter](#)

TypeConverter

Verwenden Sie den typeConverter Prozessor, um einen Wertetyp, der dem angegebenen Schlüssel zugeordnet ist, in den angegebenen Typ zu konvertieren. Es handelt sich um einen Casting-Prozessor, der die Typen der angegebenen Felder ändert. Werte können in einen der folgenden Datentypen konvertiert werden: integerdouble, string und boolean

Feld	Description	Erforderlich?	Standard	Einschränkungen
Einträge	Array von Einträgen. Jedes Element im Array muss type Felder enthaltenkey.	Ja		Maximale Anzahl von Einträgen: 10
Schlüssel	Der Schlüssel mit dem Wert, der in einen anderen Typ konvertiert werden soll	Ja		Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
type	Der Typ, in den konvertiert werden soll. Gültige Werte sind	Ja		

Feld	Description	Erforderlich?	Standard	Einschränkungen
	integerdouble, string undboolean.			

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{
  "name": "value",
  "status": "200"
}
```

Die Transformer-Konfiguration sieht wie folgt aus und verwendet `typeConverter` mit `parseJSON`:

```
[
  {
    "parseJSON": {}
  },
  {
    "typeConverter": {
      "entries": [
        {
          "key": "status",
          "type": "integer"
        }
      ]
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
  "name": "value",
  "status": 200
}
```

DateTimeConverter

Verwenden Sie den `datetimeConverter` Prozessor, um eine Datetime-Zeichenfolge in ein von Ihnen angegebenes Format zu konvertieren.

Feld	Description	Erforderlich?	Standard	Einschränkungen
Quelle	Der Schlüssel, auf den die Datumskonvertierung angewendet werden soll.	Ja		Maximale Anzahl von Einträgen: 10
Muster abgleichen	Eine Liste von Mustern, die mit dem Feld abgeglichen werden sollen <code>source</code>	Ja		Maximale Anzahl von Einträgen: 5
Ziel	Das JSON-Feld, in dem das Ergebnis gespeichert werden soll.	Ja		Höchstlänge: 128 Maximale Tiefe verschachtelter Schlüssel: 3
Zielformat	Das Datetime-Format, das für die konvertierten Daten im Zielfeld verwendet werden soll.	Nein	yyyy-MM-dd'T'HH:mm:ss.SSS'Z'	Maximale Länge: 64
Zeitzone der Quelle	Die Zeitzone des Quellfeldes. Eine Liste möglicher Werte finden Sie unter Von Java unterstützte Zonen-IDs und Offsets .	Nein	UTC	Mindestlänge:1
Ziel-Zeitzone	Die Zeitzone des Zielfeldes. Eine Liste möglicher Werte finden Sie unter Von Java unterstützte Zonen-IDs und Offsets .	Nein	UTC	Mindestlänge:1

Feld	Description	Erforderlich?	Standard	Einschränkungen
locale	Das Gebietsschema des Quellfeldes. Eine Liste möglicher Werte finden Sie unter Locale getAvailableLocales () -Methode in Java mit Beispielen.	Ja		Mindestlänge:1

Beispiel

Nehmen wir das folgende Beispiel für ein Protokollereignis:

```
{"german_datetime": "Samstag 05. Dezember 1998 11:00:00"}
```

Die Transformer-Konfiguration sieht wie folgt aus und verwendet `dateTimeConverter` mit `parseJSON`:

```
[
  {
    "parseJSON": {}
  },
  {
    "dateTimeConverter": {
      "source": "german_datetime",
      "target": "target_1",
      "locale": "de",
      "matchPatterns": ["EEEE dd. MMMM yyyy HH:mm:ss"],
      "sourceTimezone": "Europe/Berlin",
      "targetTimezone": "America/New_York",
      "targetFormat": "yyyy-MM-dd'T'HH:mm:ss z"
    }
  }
]
```

Das transformierte Protokollereignis wäre das Folgende.

```
{
```

```
"german_datetime": "Samstag 05. Dezember 1998 11:00:00",  
"target_1": "1998-12-05T17:00:00 MEZ"  
}
```

Metriken und Fehler bei der Transformation

CloudWatch Logs veröffentlicht Transformationsmetriken unter CloudWatch. Zu diesen Metriken gehören `TransformedLogEvents`, `TransformedBytes`, und `TransformationErrors`. Weitere Informationen finden Sie unter [Metriken und Dimensionen von Log Transformer](#).

Immer wenn CloudWatch Logs versucht und dabei scheitert, ein Protokollereignis zu transformieren, wird diesem Protokollereignis ein `@transformationError` Systemfeld hinzugefügt. Wenn Sie eine CloudWatch Logs Insights-Abfrage ausführen, wird dieses Feld in allen Protokollereignissen angezeigt, bei denen ein Transformationsfehler aufgetreten ist. Sie können dieses Feld mit einer Abfrage abfragen, z. B. `filter ispresent(@transformationError)` um alle fehlgeschlagenen Transformationsereignisse zu finden.

Analysieren Sie mit Amazon OpenSearch Service

CloudWatch Logs lässt sich in Logs integrieren Amazon OpenSearch Service , sodass Sie automatisch kuratierte Dashboards erstellen können, die wichtige Kennzahlen anzeigen, die OpenSearch Service aus Protokollen ableitet, die von Services verkauft wurden. AWS Die folgenden Dashboards sind verfügbar:

- Ein Amazon VPC Flow Logs-Dashboard erfasst Netzwerkflussdaten für Amazon VPC. Es hilft Ihnen, den Netzwerkverkehr zu analysieren, ungewöhnliche Muster zu erkennen und die Ressourcennutzung zu überwachen. Zu den wichtigsten angezeigten Metriken gehören die folgenden:
 - Gesamtzahl der Ströme sowie Akzeptanz und Ablehnung dieser Ströme
 - Verkehrsmuster im Laufe der Zeit
 - Ein Sankey-Diagramm, das den Datenfluss zwischen Quelle und Ziel veranschaulicht IPs (Top-Talker)
 - IPs Nach Anzahl der übertragenen Bytes und Pakete

Note

Derzeit wird nur das Feldformat VPC Version 2 unterstützt.

- Ein AWS WAF Logs-Dashboard bietet Einblicke in den Web-Traffic, der von AWS WAF überwacht wird. Dieses Dashboard hilft Ihnen dabei, Verkehrsmuster, blockierte Anfragen und potenzielle Bedrohungen aus bestimmten Regionen oder zu identifizieren IPs. Zu den wichtigsten Kennzahlen, die angezeigt werden, gehören:
 - Gesamtzahl der Anfragen, einschließlich der Anzahl der Anfragen nach „ZULASSEN“ und „BLOCKIEREN“.
 - Anforderungsverlauf im Zeitverlauf, wobei zulässige und blockierte Anfragen angezeigt werden.
 - Aufschlüsselung der Anfragen nach Web-ACL-Name, blockierten Anfragen nach Abschlussregel und Quelle. IPs
 - Eine geografische Verteilung der Herkunft der Anfragen.
 - Die wichtigsten Kunden IPs - und Abschlussregeln nach Anzahl der Anfragen.
- Ein CloudTrail Protokoll-Dashboard bietet mithilfe von CloudTrail Protokollen einen Überblick über die API-Aktivitäten in Ihrer AWS Umgebung. Es ist nützlich, um API-Aktivitäten zu überwachen,

Aktionen zu prüfen und potenzielle Sicherheits- oder Compliance-Probleme zu identifizieren. Zu den wichtigsten Kennzahlen, die angezeigt werden, gehören:

- Gesamtzahl der Ereignisse und Verlauf der Ereignisse im Zeitverlauf
- Eine Aufschlüsselung der Ereignisse nach Konto IDs, Kategorien und Regionen.
- Top APIs, Dienste und Quelle, die an der Generierung von Ereignissen IPs beteiligt waren.
- Eine Tabelle der wichtigsten Benutzer, die Ereignisse generieren, mit detaillierten Informationen zu Benutzerkonten und der Anzahl der Ereignisse.
- Ein AWS Network FirewallDashboard bietet einen besseren Einblick in den Netzwerkverkehr und bietet wertvolle Einblicke für die Sicherheitsüberwachung und -analyse. Dieses Dashboard bietet einen umfassenden Überblick über verschiedene Netzwerkmetriken und -muster, sodass potenzielle Sicherheitsprobleme schnell erkannt und Netzwerkkonfigurationen optimiert werden können. Zu den wichtigsten angezeigten Metriken gehören die folgenden:
 - Die wichtigsten Redner und Protokolle
 - Einblicke in Endgeräte PrivateLink
 - Zulässiger und blockierter Datenverkehr mit Angabe von TLS-Servernamen

Die in diesen kuratierten Dashboards angezeigten Metriken stammen aus Amazon OpenSearch Service Analysen.

Bevor Sie diese Dashboards anzeigen können, müssen Sie eine IAM-Rolle erstellen und eine einmalige Integration von Logs mit durchführen. CloudWatch Amazon OpenSearch Service Diese einmalige Integration konfiguriert die Amazon OpenSearch Service Ressourcen, die zum Erstellen und Rendern des Dashboards benötigt werden. Für die in Anspruch genommenen OpenSearch Dienste fallen Gebühren an. Weitere Informationen finden Sie unter [CloudWatch Amazon-Preise](#).

Sie können diese kuratierten Dashboards nur für Protokollgruppen in der Standard-Protokollklasse erstellen.

Important

Verwenden Sie [Log-Transformatoren](#) nicht für Log-Gruppen, für die Sie Dashboards für verkaufte Logs erstellen möchten. Die Transformation von Protokollereignissen führt dazu, dass die Dashboards leere Daten enthalten.

Themen

- [Schritt 1: Erstellen Sie die Integration mit Service OpenSearch](#)
- [Schritt 2: Erstellen Sie Dashboards für verkaufte Logs](#)
- [Dashboards für verkaufte Logs anzeigen, bearbeiten oder löschen](#)
- [IAM-Richtlinien für Benutzer](#)
- [Berechtigungen, die für die Integration erforderlich sind](#)

Schritt 1: Erstellen Sie die Integration mit Service OpenSearch

Der erste Schritt besteht darin, die Integration mit OpenSearch Service zu erstellen, was Sie nur einmal tun müssen. Durch die Erstellung der Integration werden die folgenden Ressourcen in Ihrem Konto erstellt.

- [Eine OpenSearch Service Zeitreihensammlung](#) ohne hohe Verfügbarkeit.

Eine Sammlung besteht aus einer Reihe von OpenSearch Dienstindizes, die zusammenarbeiten, um eine Arbeitslast zu unterstützen.

- Zwei Sicherheitsrichtlinien für die Sammlung. In der einen wird der Verschlüsselungstyp definiert, der entweder mit einem vom Kunden verwalteten AWS KMS Schlüssel oder mit einem Schlüssel, der dem Dienst gehört. Die andere Richtlinie definiert den Netzwerkzugriff, sodass die OpenSearch Dienstanwendung auf die Sammlung zugreifen kann. Weitere Informationen finden Sie unter [Verschlüsselung ruhender Daten für Amazon OpenSearch Service](#).
- [Eine Richtlinie für den Zugriff auf OpenSearch Servicedaten](#), die definiert, wer auf Daten in der Sammlung zugreifen kann.
- [Eine OpenSearch Service-Direktabfrage-Datenquelle](#) mit als Quelle definierten CloudWatch Protokollen.
- [Eine OpenSearch Dienstanwendung](#) mit dem Namen `aws-analytics`. Die Anwendung wird so konfiguriert, dass sie die Erstellung eines Workspace ermöglicht. Wenn eine Anwendung mit dem Namen `aws-analytics` bereits existiert, wird sie aktualisiert, um diese Sammlung als Datenquelle hinzuzufügen.
- [Ein OpenSearch Service-Workspace](#), der die Dashboards hostet und es allen Benutzern, denen Zugriff gewährt wurde, ermöglicht, aus dem Workspace zu lesen.

Themen

- [Erforderliche Berechtigungen](#)

- [Erstellen Sie die Integration](#)

Erforderliche Berechtigungen

Um die Integration zu erstellen, müssen Sie bei einem Konto angemeldet sein, das über die CloudWatchOpenSearchDashboardsFullAccessverwaltete IAM-Richtlinie oder gleichwertige Berechtigungen verfügt, wie hier gezeigt. Sie benötigen außerdem diese Berechtigungen, um die Integration zu löschen, Dashboards zu erstellen, zu bearbeiten und zu löschen und das Dashboard manuell zu aktualisieren.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "CloudWatchOpenSearchDashboardsIntegration",
    "Effect": "Allow",
    "Action": [
      "logs:ListIntegrations",
      "logs:GetIntegration",
      "logs>DeleteIntegration",
      "logs:PutIntegration",
      "logs:DescribeLogGroups",
      "opensearch:ApplicationAccessAll",
      "iam:ListRoles",
      "iam:ListUsers"
    ],
    "Resource": "*"
  },
  {
    "Sid": "CloudWatchLogsOpensearchReadAPIs",
    "Effect": "Allow",
    "Action": [
      "aoss:BatchGetCollection",
      "aoss:BatchGetLifecyclePolicy",
      "es:ListApplications"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:CalledViaFirst": "logs.amazonaws.com"
      }
    }
  }
]
```

```

    }
  }
},
{
  "Sid": "CloudWatchLogsOpensearchCreateServiceLinkedAccess",
  "Effect": "Allow",
  "Action": [
    "iam:CreateServiceLinkedRole"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/
opensearchservice.amazonaws.com/AWSServiceRoleForAmazonOpenSearchService",
  "Condition": {
    "StringEquals": {
      "iam:AWSServiceName": "opensearchservice.amazonaws.com",
      "aws:CalledViaFirst": "logs.amazonaws.com"
    }
  }
},
{
  "Sid": "CloudWatchLogsObservabilityCreateServiceLinkedAccess",
  "Effect": "Allow",
  "Action": [
    "iam:CreateServiceLinkedRole"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/
observability.aoss.amazonaws.com/AWSServiceRoleForAmazonOpenSearchServerless",
  "Condition": {
    "StringEquals": {
      "iam:AWSServiceName": "observability.aoss.amazonaws.com",
      "aws:CalledViaFirst": "logs.amazonaws.com"
    }
  }
},
{
  "Sid": "CloudWatchLogsCollectionRequestAccess",
  "Effect": "Allow",
  "Action": [
    "aoss:CreateCollection"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:CalledViaFirst": "logs.amazonaws.com",
      "aws:RequestTag/CloudWatchOpenSearchIntegration": [

```

```

        "Dashboards"
    ],
    },
    "ForAllValues:StringEquals": {
        "aws:TagKeys": "CloudWatchOpenSearchIntegration"
    }
},
{
    "Sid": "CloudWatchLogsApplicationRequestAccess",
    "Effect": "Allow",
    "Action": [
        "es:CreateApplication"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:CalledViaFirst": "logs.amazonaws.com",
            "aws:RequestTag/OpenSearchIntegration": [
                "Dashboards"
            ]
        },
        "ForAllValues:StringEquals": {
            "aws:TagKeys": "OpenSearchIntegration"
        }
    }
},
{
    "Sid": "CloudWatchLogsCollectionResourceAccess",
    "Effect": "Allow",
    "Action": [
        "aoss:DeleteCollection"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:CalledViaFirst": "logs.amazonaws.com",
            "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                "Dashboards"
            ]
        }
    }
},
{

```



```

        "Sid": "CloudWatchLogsApplicationResourceAccess",
        "Effect": "Allow",
        "Action": [
            "es:UpdateApplication",
            "es:GetApplication"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:CalledViaFirst": "logs.amazonaws.com",
                "aws:ResourceTag/OpenSearchIntegration": [
                    "Dashboards"
                ]
            }
        }
    },
    {
        "Sid": "CloudWatchLogsCollectionPolicyAccess",
        "Effect": "Allow",
        "Action": [
            "aoss:CreateSecurityPolicy",
            "aoss:CreateAccessPolicy",
            "aoss>DeleteAccessPolicy",
            "aoss>DeleteSecurityPolicy",
            "aoss:GetAccessPolicy",
            "aoss:GetSecurityPolicy"
        ],
        "Resource": "*",
        "Condition": {
            "StringLike": {
                "aoss:collection": "cloudwatch-logs-*",
                "aws:CalledViaFirst": "logs.amazonaws.com"
            }
        }
    },
    {
        "Sid": "CloudWatchLogsAPIAccessAll",
        "Effect": "Allow",
        "Action": [
            "aoss:APIAccessAll"
        ],
        "Resource": "*",
        "Condition": {
            "StringLike": {

```

```

        "aoss:collection": "cloudwatch-logs-*"
    }
}
},
{
    "Sid": "CloudWatchLogsIndexPolicyAccess",
    "Effect": "Allow",
    "Action": [
        "aoss:CreateAccessPolicy",
        "aoss:DeleteAccessPolicy",
        "aoss:GetAccessPolicy",
        "aoss:CreateLifecyclePolicy",
        "aoss:DeleteLifecyclePolicy"
    ],
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "aoss:index": "cloudwatch-logs-*",
            "aws:CalledViaFirst": "logs.amazonaws.com"
        }
    }
},
{
    "Sid": "CloudWatchLogsDQSRequestQueryAccess",
    "Effect": "Allow",
    "Action": [
        "es:AddDirectQueryDataSource"
    ],
    "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*",
    "Condition": {
        "StringEquals": {
            "aws:CalledViaFirst": "logs.amazonaws.com",
            "aws:RequestTag/CloudWatchOpenSearchIntegration": [
                "Dashboards"
            ]
        },
        "ForAllValues:StringEquals": {
            "aws:TagKeys": "CloudWatchOpenSearchIntegration"
        }
    }
},
{
    "Sid": "CloudWatchLogsStartDirectQueryAccess",
    "Effect": "Allow",

```

```

        "Action": [
            "opensearch:StartDirectQuery",
            "opensearch:GetDirectQuery"
        ],
        "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*"
    },
    {
        "Sid": "CloudWatchLogsDQSResourceQueryAccess",
        "Effect": "Allow",
        "Action": [
            "es:GetDirectQueryDataSource",
            "es>DeleteDirectQueryDataSource"
        ],
        "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*",
        "Condition": {
            "StringEquals": {
                "aws:CalledViaFirst": "logs.amazonaws.com",
                "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                    "Dashboards"
                ]
            }
        }
    },
    {
        "Sid": "CloudWatchLogsPassRoleAccess",
        "Effect": "Allow",
        "Action": [
            "iam:PassRole"
        ],
        "Resource": "*",
        "Condition": {
            "StringLike": {
                "iam:PassedToService":
"directquery.opensearchservice.amazonaws.com",
                "aws:CalledViaFirst": "logs.amazonaws.com"
            }
        }
    },
    {
        "Sid": "CloudWatchLogsAossTagsAccess",
        "Effect": "Allow",
        "Action": [
            "aoss:TagResource"
        ],

```

```

    "Resource": "arn:aws:aoss:*:*:collection/*",
    "Condition": {
      "StringEquals": {
        "aws:CalledViaFirst": "logs.amazonaws.com",
        "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
          "Dashboards"
        ]
      },
      "ForAllValues:StringEquals": {
        "aws:TagKeys": "CloudWatchOpenSearchIntegration"
      }
    }
  },
  {
    "Sid": "CloudWatchLogsEsApplicationTagsAccess",
    "Effect": "Allow",
    "Action": [
      "es:AddTags"
    ],
    "Resource": "arn:aws:opensearch:*:*:application/*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/OpenSearchIntegration": [
          "Dashboards"
        ],
        "aws:CalledViaFirst": "logs.amazonaws.com"
      },
      "ForAllValues:StringEquals": {
        "aws:TagKeys": "OpenSearchIntegration"
      }
    }
  },
  {
    "Sid": "CloudWatchLogsEsDataSourceTagsAccess",
    "Effect": "Allow",
    "Action": [
      "es:AddTags"
    ],
    "Resource": "arn:aws:opensearch:*:*:datasource/*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
          "Dashboards"
        ]
      }
    ]
  },

```

```
        "aws:CalledViaFirst": "logs.amazonaws.com"
      },
      "ForAllValues:StringEquals": {
        "aws:TagKeys": "CloudWatchOpenSearchIntegration"
      }
    }
  }
}
```

Erstellen Sie die Integration

Gehen Sie wie folgt vor, um die Integration zu erstellen.

Um CloudWatch Logs zu integrieren mit Amazon OpenSearch Service

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im linken Navigationsbereich Logs Insights und dann den OpenSearch Tab Analysieren mit aus.
3. Wählen Sie „Integration erstellen“.
4. Geben Sie für Integrationsname einen Namen für die Integration ein.
5. (Optional) Um die in OpenSearch Service Serverless geschriebenen Daten zu verschlüsseln, geben Sie den ARN des AWS KMS Schlüssels ein, den Sie in den KMS-Schlüssel-ARN verwenden möchten. Weitere Informationen finden Sie unter [Verschlüsselung im Ruhezustand](#) im Amazon OpenSearch Service Developer Guide.
6. Geben Sie für die Datenaufbewahrung den Zeitraum ein, für den die OpenSearch Service-Datenindizes aufbewahrt werden sollen. Dies definiert auch den maximalen Zeitraum, für den Sie Daten in den Dashboards anzeigen können. Wenn Sie eine längere Datenaufbewahrungsdauer wählen, fallen zusätzliche Such- und Indexierungskosten an. Weitere Informationen finden Sie unter [OpenSearch Service Serverless Pricing](#).

Die maximale Aufbewahrungsfrist beträgt 30 Tage.

Die Dauer der Datenspeicherung wird auch zur Erstellung der Lebenszyklusrichtlinie für die OpenSearch Servicesammlung verwendet.

7. Erstellen Sie für die IAM-Rolle für das Schreiben in die OpenSearch Sammlung eine neue IAM-Rolle oder wählen Sie eine vorhandene IAM-Rolle aus, die zum Schreiben in die OpenSearch Servicesammlung verwendet werden soll.

Das Erstellen einer neuen Rolle ist die einfachste Methode, und die Rolle wird mit den erforderlichen Berechtigungen erstellt.

 Note

Wenn Sie eine Rolle erstellen, verfügt sie über Leserechte aus allen Protokollgruppen im Konto.

Wenn Sie eine bestehende Rolle auswählen möchten, sollte sie über die unter aufgeführten Berechtigungen verfügen [Berechtigungen, die für die Integration erforderlich sind](#).

Alternativ können Sie „Bestehende Rolle verwenden“ auswählen und dann im Abschnitt „Zugriffsberechtigungen der ausgewählten Rolle überprüfen“ die Option „Rolle erstellen“ auswählen. Auf diese Weise können Sie die unter aufgeführten Berechtigungen [Berechtigungen, die für die Integration erforderlich sind](#) als Vorlage verwenden und diese ändern. Dies ist beispielsweise der Fall, wenn Sie eine genauere Steuerung der Protokollgruppen angeben möchten.

8. Für IAM-Rollen und Benutzer, die Dashboards anzeigen können, wählen Sie aus, wie Sie IAM-Rollen und IAM-Benutzern Zugriff auf das Dashboard mit verkauften Protokollen gewähren möchten:
 - Um den Dashboard-Zugriff auf einige Benutzer zu beschränken, wählen Sie IAM-Rollen und Benutzer auswählen, die Dashboards anzeigen können. Suchen Sie dann im Textfeld nach den IAM-Rollen und IAM-Benutzern, denen Sie Zugriff gewähren möchten, und wählen Sie diese aus.
 - Um allen Benutzern Zugriff auf das Dashboard zu gewähren, wählen Sie Allen Rollen und Benutzern in diesem Konto das Anzeigen von Dashboards erlauben.

 Important

Wenn Sie Rollen oder Benutzer oder alle Benutzer auswählen, werden diese nur der [Datenzugriffsrichtlinie hinzugefügt, die für den Zugriff](#) auf die OpenSearch Service-Sammlung erforderlich ist, in der die Dashboard-Daten gespeichert werden. Damit sie die Dashboards der verkauften Logs einsehen können, müssen Sie

diesen Rollen und Benutzern auch die [CloudWatchOpenSearchDashboardAccess](#) verwaltete IAM-Richtlinie zuweisen.

9. Wählen Sie „Integration erstellen“

Das Erstellen der Integration dauert einige Minuten.

Schritt 2: Erstellen Sie Dashboards für verkaufte Logs

Nachdem Sie die Integration erstellt haben, können Sie Dashboards erstellen. Dashboards sind für Amazon VPC-Flow-Logs, Logs und CloudTrail AWS WAF Logs verfügbar.

Um ein verkäufliches Log-Dashboard mit von Service abgeleiteten Metriken zu erstellen OpenSearch

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im linken Navigationsbereich Logs Insights und dann den OpenSearch Tab Analysieren mit aus.
3. Klicken Sie auf Dashboard erstellen.
4. Wählen Sie aus, für welche Art von Protokollen das Dashboard erstellt werden soll, AWS WAF, Amazon VPC-Flow-Logs, CloudTrail, oder AWS Network Firewall.
5. Geben Sie einen Namen für das Dashboard und optional eine Beschreibung ein.
6. Geben Sie unter Häufigkeit der Datensynchronisierung ein, wie oft OpenSearch Service Abfragen durchführen soll, CloudWatch damit die in OpenSearch Service erstellten Metriken und Indizes synchronisiert und mit neuen Daten aktualisiert werden können. OpenSearch Service erstellt Metriken und Indizes in Ihren Protokollen, um das Dashboard zu rendern.

Wenn Sie eine kürzere Zeit wählen, bleiben die Daten auf dem neuesten Stand und es fallen höhere Kosten an.

7. Wählen Sie die Protokollgruppen aus, aus denen Daten für dieses Dashboard gesammelt werden sollen. Achten Sie darauf, Protokollgruppen auszuwählen, die dem Dashboard-Typ entsprechen, den Sie erstellen.

Sie können die Schaltfläche Protokollgruppen durchsuchen und die Option Protokollbeispiele aus ausgewählten Protokollgruppen anzeigen verwenden, um sicherzustellen, dass Sie die gewünschten Protokollgruppen erhalten.

8. Klicken Sie auf Dashboard erstellen.

Das Dashboard wird zunächst ohne Daten angezeigt. Nach einigen Minuten werden Daten im Dashboard angezeigt. Wenn die Daten zum ersten Mal angezeigt werden, beziehen sie sich auf die Logeinträge der letzten 15 Minuten.

Dashboards für verkaufte Logs anzeigen, bearbeiten oder löschen

Dashboards für verkaufte Logs in Logs or Service anzeigen CloudWatch OpenSearch

Um Dashboards anzeigen zu können, müssen Sie bei einem IAM-Principal angemeldet sein, für den die IAM-Richtlinie gilt. `CloudWatchOpenSearchDashboardAccess`

Um die Log-Dashboards von Vended zu sehen

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im linken Navigationsbereich Logs Insights und dann den OpenSearch Tab Analysieren mit aus.
3. Wählen Sie das Dashboard im Feld „OpenSearch Dashboards“ aus.
4. (Optional) Wählen Sie oben rechts die Option Anzeigen in OpenSearch.

Die OpenSearch Servicekonsole wird geöffnet und Sie sehen dort dasselbe Dashboard. In der OpenSearch Servicekonsole können Sie Änderungen am Dashboard und seinen Widgets vornehmen. Diese Änderungen sind auch sichtbar, wenn Sie das Dashboard unter CloudWatch Logs aufrufen.

Gewähren Sie zusätzlichen IAM-Rollen oder IAM-Benutzern Zugriff auf das Dashboard

Gehen Sie wie folgt vor, um zusätzlichen IAM-Prinzipalen Zugriff zu gewähren, nachdem Sie die Integration erstellt haben.

Um zusätzlichen IAM-Rollen oder -Benutzern Zugriff auf das angebotene Log-Dashboard zu gewähren

1. Bearbeiten Sie die Datenzugriffsrichtlinie für die Sammlung, um diese Rollen oder Benutzer hinzuzufügen. Weitere Informationen finden Sie unter [Datenzugriffskontrolle für Amazon OpenSearch Service Serverless](#) im OpenSearch Service Developer Guide.
2. Gewähren Sie CloudWatchOpenSearchDashboardAccess diesen Benutzern das. Weitere Informationen zum Inhalt dieser Richtlinie finden Sie unter [CloudWatchOpenSearchDashboardAccess](#).

Dashboard-Konfiguration bearbeiten

Sie können den Namen, die Beschreibung und die Synchronisierungshäufigkeit vorhandener Verkaufsprotokoll-Dashboards bearbeiten.

Um ein Dashboard für ausgelieferte Logs zu bearbeiten

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im linken Navigationsbereich Logs Insights und dann den OpenSearch Tab Analysieren mit aus.
3. Wählen Sie das Dashboard im Feld „OpenSearch Dashboards“ aus.
4. Wählen Sie „Aktionen“, „Dashboard-Details ändern“.
5. Nehmen Sie Ihre Änderungen vor und wählen Sie dann Änderungen bestätigen.

Lösche ein ausgeliefertes Log-Dashboard

Sie können ein Verkaufs-Log-Dashboard löschen. Wenn Sie dies tun, werden das Dashboard, die Metriken und Indizes, die in der OpenSearch Service-Sammlung erstellt wurden, alle gelöscht.

Note

Warten Sie nach dem Löschen eines ausgelieferten Log-Dashboards mindestens sechs Stunden, bevor Sie versuchen, dasselbe Dashboard erneut zu erstellen. Wenn Sie nicht warten, funktioniert das neu erstellte Dashboard nicht richtig.

Um ein verkauftes Log-Dashboard zu löschen

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im linken Navigationsbereich Logs Insights und dann den OpenSearch Tab Analysieren mit aus.
3. Wählen Sie das Dashboard im Feld „OpenSearch Dashboards“ aus.
4. Wählen Sie Aktionen, Löschen aus.
5. Bestätigen Sie Ihre Entscheidung durch Eingabe **delete** und wählen Sie dann Löschen.

Löschen Sie alle verkauften Log-Dashboard-Integrationen mit Service OpenSearch

Sie können Ihre gesamte OpenSearch Integration löschen. Wenn Sie dies tun, werden alle versendeten Log-Dashboards und die darin angezeigten Daten gelöscht.

Important

Um laufende Kosten zu vermeiden, empfehlen wir dringend, die folgenden Ressourcen manuell zu löschen, bevor Sie die Integration löschen. Durch das Löschen der Integration werden diese Ressourcen nicht automatisch gelöscht. Nachdem Sie die Integration gelöscht haben, können Sie nicht mehr auf diese Ressourcen zugreifen, um sie zu löschen. Gehen Sie wie folgt vor, um die Namen der zu löschenden Ressourcen zu ermitteln.

- [Die Datenquelle](#)
- [Die Sammlung](#)
- [Die Datenzugriffsrichtlinie](#)
- [Die Verschlüsselungsrichtlinie](#)
- [Die Netzwerkrichtlinie](#)
- [Die Lebenszyklus-Richtlinie](#)

Um Ihre gesamte Log-Dashboard-Integration mit OpenSearch Service zu löschen

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im linken Navigationsbereich die Option Einstellungen aus.

3. Wählen Sie die Registerkarte Protokolle aus.
4. Wählen Sie im OpenSearch Integrationsbereich die Option Integration löschen aus.

Auf dem nächsten Bildschirm werden die Namen der OpenSearch Servicere Ressourcen angezeigt, die Sie löschen sollten, bevor Sie die Integration löschen.

5. Bestätigen Sie Ihre Entscheidung **delete**, indem Sie die Eingabe eingeben und dann Integration löschen wählen.

IAM-Richtlinien für Benutzer

CloudWatch Logs hat zwei IAM-Richtlinien erstellt, und CloudWatchOpenSearchDashboardsFullAccess. CloudWatchOpenSearchDashboardAccess In der folgenden Tabelle ist aufgeführt, welche Aktionen jede dieser Richtlinien ermöglicht.

Action	IAM-Richtlinie	Zusätzliche Berechtigungen erforderlich
Integration erstellen	CloudWatchOpenSearchDashboardsFullAccess	
Integration löschen	CloudWatchOpenSearchDashboardsFullAccess	
Dashboard erstellen	CloudWatchOpenSearchDashboardsFullAccess	
Dashboard bearbeiten	CloudWatchOpenSearchDashboardsFullAccess	
Dashboard löschen	CloudWatchOpenSearchDashboardsFullAccess	
Aktualisieren Sie das Dashboard mithilfe von Synchronize now	CloudWatchOpenSearchDashboardsFullAccess	
Sehen Sie sich die Integration in den Einstellungen an	CloudWatchOpenSearchDashboardAccess oder	

Action	IAM-Richtlinie	Zusätzliche Berechtigungen erforderlich
	CloudWatchOpenSearchDashboardsFullAccess	
Dashboard anzeigen	CloudWatchOpenSearchDashboardAccess oder CloudWatchOpenSearchDashboardsFullAccess	Geben Sie die Rolle oder den Benutzer an, wenn Sie die Integration erstellen , oder bearbeiten Sie die Datenzugriffsrichtlinie für die Sammlung, um diese Rollen oder Benutzer hinzuzufügen. Weitere Informationen finden Sie unter Datenzugriffskontrolle für Amazon OpenSearch Service Serverless im OpenSearch Service Developer Guide.
Dashboard in der OpenSearch Service-Konsole anzeigen	CloudWatchOpenSearchDashboardAccess oder CloudWatchOpenSearchDashboardsFullAccess	Geben Sie die Rolle oder den Benutzer an, wenn Sie die Integration erstellen , oder bearbeiten Sie die Datenzugriffsrichtlinie für die Sammlung, um diese Rollen oder Benutzer hinzuzufügen. Weitere Informationen finden Sie unter Datenzugriffskontrolle für Amazon OpenSearch Service Serverless im OpenSearch Service Developer Guide.

Berechtigungen, die für die Integration erforderlich sind

Wenn Sie eine IAM-Rolle erstellen, die von der Integration verwendet werden soll, muss sie die folgenden Berechtigungen und Vertrauensrichtlinien enthalten, anstatt CloudWatch Logs die Erstellung der Rolle zu gestatten. Weitere Informationen zum Erstellen einer IAM-Rolle finden Sie unter [Eine Rolle erstellen, um Berechtigungen an einen Dienst zu delegieren](#). AWS

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchLogsAccess",
      "Effect": "Allow",
      "Action": [
        "logs:StartQuery",
        "logs:GetLogGroupFields",
        "logs:GetQueryResults"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "CloudWatchLogsDescribeLogGroupsAccess",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AmazonOpenSearchCollectionAccess",
      "Effect": "Allow",
      "Action": [
        "aoss:APIAccessAll"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aoss:collection": "cloudwatch-logs-*"
        }
      }
    }
  ]
}
```

```
}  
}  
}  
]  
}
```

Note

Die vorherige Rolle gewährt Lesezugriff aus allen Protokollgruppen im Konto, sodass Sie Dashboards für jedes Protokollkonto, einschließlich kontoübergreifender Protokollgruppen, erstellen können. Wenn Sie den Zugriff auf bestimmte Protokollgruppen einschränken und Dashboards nur für diese Protokollgruppen erstellen möchten, können Sie die erste Aussage in dieser Richtlinie wie folgt aktualisieren:

```
{  
  "Sid": "CloudWatchLogsAccess",  
  "Effect": "Allow",  
  "Action": [  
    "logs:StartQuery",  
    "logs:GetLogGroupFields",  
    "logs:GetQueryResults"  
  ],  
  "Resource": [  
    "arn:aws:logs:us-east-1:123456789012:log-group:myLogGroup:*",  
    "arn:aws:logs:us-east-1:123456789012:log-group:myLogGroup"  
  ]  
}
```

Zugriffsprotokolle mit der Integration von S3-Tabellen

Die Integration von S3 Tables mit CloudWatch ermöglicht Ihnen den Zugriff auf Protokolldaten, die CloudWatch mithilfe von Analyse-Engines wie Amazon Athena, Amazon Redshift und Tools von Drittanbietern aufgenommen wurden, die die Verbindung zu Apache-Speichern unterstützen. Iceberg-compatible Diese Integration ermöglicht es Ihnen, umfassende Protokollanalysen mit Tools Ihrer Wahl durchzuführen und Daten in CloudWatch Logs mit Nichtdaten zu korrelieren. CloudWatch

Die Integration von S3-Tabellen verstehen

Amazon S3 Tables Integration ist eine vollständig verwaltete Lösung, die Ihre Logs in CloudWatch Logs als verwaltete Amazon S3 S3-Tabellen verfügbar macht. Mit dieser Integration erhalten Sie zusätzlich zu den Protokollfunktionen mehr Flexibilität bei der Analyse Ihrer CloudWatch Protokolle.

Die Integration funktioniert durch die Erstellung eines verwalteten Amazon S3 S3-Tabellen-Buckets (`aws-cloudwatch`) und das Zuordnen bestimmter Protokollquellen zu Amazon S3 S3-Tabellen auf der Grundlage von Name und Typ der Datenquelle (die über die Registerkarte Protokollverwaltung > Datenquellen in der CloudWatch Logs Console verwaltet werden können). Nach der Verknüpfung können CloudWatch Logs-Daten über Amazon S3 S3-Tabellen im Apache Iceberg-Format abgerufen werden. Dieses Format bietet eine standardisierte Möglichkeit für verschiedene Analyse-Engines, die Daten effizient abzufragen. Diese Funktion ist ohne zusätzliche Kosten verfügbar.

Kernkomponenten

Apache Iceberg-Tabellen

Das von S3 Tables verwendete zugrunde liegende Tabellenformat, das strukturierte Datenspeicher bietet und die Kompatibilität mit mehreren Analyse-Engines ermöglicht.

Zuordnung der Datenquellen

Der Prozess der Verknüpfung bestimmter CloudWatch Protokollquellen mit der Integration von S3-Tabellen auf der Grundlage von Datenquellen- und Typkriterien.

Datenfluss zu S3-Tabellen

Wenn Sie wissen, wie Daten zwischen CloudWatch Protokollen und S3-Tabellen fließen, können Sie Ihre Integration planen und Ihre Protokolldaten effektiv verwalten.

Wenn Sie eine Zuordnung erstellen, übermittelt CloudWatch Logs automatisch neue Protokollereignisse, die dem Namen und Typ der zugehörigen Datenquelle entsprechen, an einen CloudWatch-managed S3-Tabellen-Bucket. Diese Ereignisse werden im Logs-Namespace unter der entsprechenden Tabelle für diese Datenquelle angezeigt. Die Integration verarbeitet nur Protokollereignisse, die nach der Erstellung der Zuordnung empfangen wurden. Bestehende Protokolldaten werden nicht aufgefüllt. Wenn Sie eine S3-Tabellen-Integration erstellen und das Kontrollkästchen Alle Protokollquellen und Typen in der S3-Tabelle verfügbar machen aktiviert lassen, werden standardmäßig alle Datenquellen in Ihrem Konto automatisch zugeordnet und an S3-Tabellen übermittelt, einschließlich aller Datenquellen, die in future hinzugefügt werden. Um nur bestimmte Datenquellen für S3-Tabellen bereitzustellen, deaktivieren Sie dieses Kontrollkästchen bei der Erstellung der Integration und ordnen Sie dann die Datenquellen, die Sie einbeziehen möchten, einzeln zu.

Die Datenspeicherung im S3-Tabellen-Bucket entspricht der Aufbewahrungsrichtlinie, die für die Protokollgruppe festgelegt wurde. Wenn Sie beispielsweise für eine Protokollgruppe die Aufbewahrung von einem Tag festlegen, entfernt CloudWatch Logs die Daten nach einem Tag sowohl aus den CloudWatch Protokollen als auch aus der S3-Tabelle. Wenn Sie eine Protokollgruppe oder einen Protokollstream löschen, entfernt CloudWatch Logs auch die Daten aus dem S3-Tabellen-Bucket.

Wann sollte die Integration von S3-Tabellen verwendet werden

Erwägen Sie die Verwendung der S3-Tabellen-Integration, um Protokolldaten mit anderen externen oder CloudWatch Nichtdaten zu korrelieren, oder wenn Sie es vorziehen, andere Analysetools wie Amazon Athena für die Durchführung von Analysen von CloudWatch Protokolldaten zu verwenden. Verwenden Sie diese Integration, wenn Sie Funktionen benötigen, die über das hinausgehen, was in CloudWatch Logs verfügbar ist. Diese Integration ist besonders nützlich, wenn:

- Sie müssen komplexe SQL-like Abfragen für große Mengen von Protokolldaten ausführen
- Sie möchten die Protokollanalyse in bestehende Analyse-Workflows und -Tools integrieren
- Sie benötigen umfassende Protokollanalysefunktionen, die sich über mehrere Datenquellen erstrecken

Für S3-Tabellen, die mit dieser Integration erstellt wurden, fallen keine zusätzlichen Speicher- oder Tabellenverwaltungsgebühren an, abgesehen von den CloudWatch bestehenden Preisen für Aufnahme und Speicherung.

Voraussetzungen

Stellen Sie vor der Implementierung der Integration sicher, dass Sie über Folgendes verfügen:

- Bestehende CloudWatch Protokolldaten
- Geeignete IAM-Berechtigungen für den dienstübergreifenden Zugriff zwischen CloudWatch Protokollen und S3-Tabellen, wie im folgenden Abschnitt beschrieben

IAM-Berechtigungen

Um CloudWatch Logs in S3-Tabellen zu integrieren, müssen Sie IAM-Berechtigungen für zwei separate Entitäten konfigurieren: den Benutzer oder die Rolle, die die Integration einrichtet, und die Servicerolle, die CloudWatch Logs für das Schreiben von Daten in S3-Tabellen annimmt.

Für die Rolle oder den Benutzer, der die Integration erstellt

Der Benutzer oder die Rolle, der die Integration einrichtet, benötigt die folgenden Berechtigungen:

- `observabilityadmin:CreateS3TableIntegration`um die Integration `logs:AssociateSourceToS3TableIntegration` zu erstellen und Quellen hinzuzufügen
- `s3tables:CreateTableBuckets``s3tables:PutTableBucketEncryption`, und `s3tables:PutTableBucketPolicy` um den S3-Tabellen-Bucket zu konfigurieren

Für die Servicerolle

Hängen Sie die folgende IAM-Richtlinie an die IAM-Dienstrolle an, die CloudWatch Logs verwendet, um Daten in den Tabellen-Bucket zu schreiben. Diese Richtlinie gewährt die Berechtigung, in die Tabellen zu schreiben. Ersetzen Sie *aws-region123456789012*, und *log-group-name* durch Ihre AWS Region, Konto-ID und den Namen der Protokollgruppe.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:integrateWithS3Table"
      ],
    }
  ],
}
```

```

        "Resource": ["arn:aws:logs:aws-region:123456789012:log-group:log-group-
name"],
        "Condition": {
            "StringEquals": {
                "aws:ResourceAccount": "123456789012"
            }
        }
    }
]
}

```

Fügen Sie die folgende Vertrauensrichtlinie der IAM-Dienstrolle hinzu, von der CloudWatch Logs annimmt, dass sie Protokolldaten in S3-Tabellen schreibt. Sie erstellen oder wählen diese Rolle während des Integrations-Setups aus. Die Bedingungen schränken die Rolle so ein, dass CloudWatch Logs sie nur für das angegebene Konto und die angegebene Protokollgruppe übernehmen kann. Ersetzen Sie *aws-region123456789012*, und *log-group-name* durch Ihre AWS Region, Konto-ID und den Namen der Protokollgruppe.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": ["arn:aws:logs:aws-region:123456789012:log-
group:log-group-name"]
        }
      }
    }
  ]
}

```

KMS-Schlüsselrichtlinie (für verschlüsselte Daten)

Wenn Sie einen vom Kunden verwalteten Schlüssel zum Verschlüsseln Ihrer Protokolldaten verwenden, müssen Sie dem CloudWatch Service Principal und dem Wartungsservice-Principal von S3 Tables Zugriff auf den Schlüssel gewähren. Fügen Sie Ihrer KMS-Schlüsselrichtlinie die folgenden Anweisungen hinzu. Ersetzen Sie die Platzhalterwerte durch Ihre AWS-Konto ID, Region, KMS-Schlüssel-ID und den ARN für die S3-Tabelle oder den Tabellen-Bucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnableSystemTablesKeyUsage",
      "Effect": "Allow",
      "Principal": {
        "Service": "systemtables.cloudwatch.amazonaws.com"
      },
      "Action": [
        "kms:DescribeKey",
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:aws-region:123456789012:key/key-id",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    },
    {
      "Sid": "EnableKeyUsage",
      "Effect": "Allow",
      "Principal": {
        "Service": "maintenance.s3tables.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:aws-region:123456789012:key/key-id",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn": "<table-or-table-bucket-arn>/*"
        }
      }
    }
  ]
}
```

```
}  
  }  
    }  
      ]  
        }  
          }
```

Erste Schritte

Um mit der Integration von S3-Tabellen zu beginnen, müssen Sie die Integration zwischen Ihren CloudWatch Logs und S3-Tabellen einrichten. Dieser Prozess beinhaltet die Konfiguration von Datenquellenzuordnungen und die Einrichtung entsprechender IAM-Berechtigungen.

Um eine S3-Tabellenintegration zu erstellen

1. Öffnen Sie die CloudWatch Logs-Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie „Einstellungen“, „Global“ und „S3-Tabellenintegration erstellen“.
3. Passen Sie an, wie Logs in S3-Tabellen verschlüsselt werden und welche Rolle CloudWatch Logs beim Schreiben Ihrer Logs in S3-Tabellen spielen soll.
4. Wenn Sie möchten, dass alle Datenquellen automatisch mit der Integration verknüpft werden, lassen Sie das Kontrollkästchen Alle Protokollquellen und -typen in der S3-Tabelle verfügbar machen aktiviert (es ist standardmäßig aktiviert). Wenn Sie nur bestimmte Datenquellen zuordnen möchten, deaktivieren Sie dieses Kontrollkästchen.
5. Wählen Sie Create S3 Table Integration.

Note

Wenn Sie bei der Erstellung die Option Alle Protokollquellen und -typen in der S3-Tabelle verfügbar machen ausgewählt haben, werden alle Datenquellen automatisch verknüpft, einschließlich aller Datenquellen, die in future hinzugefügt werden. Sie können hier aufhören. Die folgenden Schritte sind nur erforderlich, wenn Sie das Kontrollkästchen deaktiviert haben und bestimmte Datenquellen zuordnen möchten.

Um Quellen einer S3-Tabellenintegration zuzuordnen

1. Öffnen Sie die CloudWatch Logs-Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie „Einstellungen“, „Global“, „S3-Tabellenintegration verwalten“.

3. Wählen Sie Datenquelle zuordnen aus.
4. Wählen Sie den Datenquellennamen und den Datenquellentyp aus, für den Sie die Integration aktivieren möchten.
5. Wählen Sie Datenquelle zuordnen aus.

Um Quellen über die Log-Management-Seite einer S3-Tabellenintegration zuzuordnen

1. Öffnen Sie die CloudWatch Logs-Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich die Option Protokollverwaltung aus.
3. Wählen Sie die Registerkarte Datenquellen aus.
4. Wählen Sie den Datenquellennamen und den Datenquellentyp aus, den Sie integrieren möchten.
5. Wählen Sie Datenquellenaktionen aus.
6. Wählen Sie Mit S3-Tabellenintegration verknüpfen aus.
7. Überprüfen Sie die Datenquellen und wählen Sie dann Datenquelle zuordnen aus.

Bevor Sie die Daten verwenden können, müssen Sie die folgenden 3 Schritte ausführen:

1. Integrieren von Amazon Amazon S3-Tabellen mit AWS Analysediensten — Verwenden der Amazon S3 S3-Konsole
2. Lake Formation Formation-Berechtigungen konfigurieren
3. Mit Analysetools Connect

Integration von Amazon Amazon S3 S3-Tabellen mit AWS Analysedienste — Verwenden der Amazon S3 S3-Konsole ([Link](#))

Um die Integration von S3-Tabellen mithilfe der S3-Konsole zu aktivieren

1. Öffnen Sie die Amazon S3 S3-Konsole unter <https://console.aws.amazon.com/s3/>.
2. Wählen Sie im linken Navigationsbereich Tabellen-Buckets aus.
3. Wählen Sie oben die Option Integration aktivieren aus.
4. Wenn Sie Tabellen-Buckets zum ersten Mal in einer beliebigen Region integrieren, erstellt Amazon Amazon S3 in Ihrem Namen eine neue IAM-Servicerolle. Diese Rolle ermöglicht Lake Formation den Zugriff auf alle Tabellen-Buckets in Ihrem Konto und den Verbundzugriff auf Ihre Tabellen in AWS Glue Data Catalog.

Lake Formation Formation-Berechtigungen konfigurieren

CloudWatch Logs ist zwar berechtigt, in die Tabelle zu schreiben (wie in den vorherigen Schritten konfiguriert), Benutzer und Analyserollen sind jedoch nicht automatisch berechtigt, die Daten zu lesen. Sie müssen den Zugriff über AWS Lake Formation ausdrücklich gewähren. Sie müssen dies für jeden IAM-Prinzipal tun, dem Sie Zugriff auf die Tabelle gewähren möchten.

Um Benutzern oder Rollen Abfragezugriff zu gewähren

Sie müssen den IAM-Prinzipalen (Benutzern oder Rollen), die Abfragen in Athena oder Redshift ausführen, die Berechtigungen SELECT und DESCRIBE gewähren.

1. Öffnen Sie die AWS Lake Formation Formation-Konsole.
2. Wählen Sie im Navigationsbereich unter Berechtigungen die Option Data Lake-Berechtigungen aus.
3. Wählen Sie Grant (Erteilen).
4. Principals: Wählen Sie die IAM-Benutzer oder -Rollen aus, für die Zugriff erforderlich ist (z. B. Ihre Datenanalysten oder die Administratorrolle, die Sie derzeit verwenden).
5. LF-Tags oder Katalogressourcen: Wählen Sie Benannte Datenkatalogressourcen aus.
6. Datenbanken und Tabellen:
 - Wählen Sie den durch die CloudWatch Integration erstellten S3-Tabellen-Bucket aus (aws-cloudwatch).
 - Wählen Sie die spezifische Tabelle aus, die Ihrer Datenquelle zugeordnet ist (optional).
7. Tabellenberechtigungen: Wählen Sie „Auswählen und beschreiben“ aus.
8. Wählen Sie Grant (Erteilen).

Note

Wenn bei der Abfrage von Logs in Athena die Fehlermeldung „Zugriff verweigert“ auftritt, stellen Sie sicher, dass der Benutzer, der die Abfrage ausführt, sowohl über die entsprechenden IAM-Berechtigungen für Athena als auch über die oben definierten Lake Formation Formation-Berechtigungen verfügt.

Weitere Informationen zu den Genehmigungen für Lake Formation finden Sie unter <https://docs.aws.amazon.com/lake-formation/latest/dg/granting-catalog-permissions.html>.

Mit Analysetools Connect

Sobald die Berechtigungen erteilt wurden, können Sie Ihren bevorzugten Analysedienst für die Abfrage der S3-Tabellen konfigurieren. S3-Tabellen verwenden das Apache Iceberg-Format, das von Amazon Athena, Amazon Redshift und Amazon EMR nativ unterstützt wird.

Um Protokolldaten in Amazon Athena abzufragen

Amazon Athena interagiert mit S3-Tabellen über den Amazon S3 Tables-Katalog.

Um Athena für die Abfrage Ihrer Logdaten einzurichten

1. Öffnen Sie die Amazon Athena Athena-Konsole unter <https://console.aws.amazon.com/athena/>.
2. Wählen Sie im Abfrage-Editor den Amazon S3 Tables-Katalog aus der Datenquellen-Dropdownliste aus.
3. Wenn Sie den Katalog nicht sehen, stellen Sie sicher, dass Sie die oben genannten Schritte zur Genehmigung von Lake Formation für Ihre spezifische Benutzerrolle abgeschlossen haben.
4. Sobald der Katalog ausgewählt ist, werden Ihre Protokolllisten in der Datenbankliste angezeigt. Sie können jetzt Standard-SQL-Abfragen für Ihre Protokolldaten ausführen.

Beispielabfrage: `SELECT * FROM "amazon_vpc__flow" LIMIT 100;`

Weitere Informationen zur Verbindung von Analytics-Diensten mit S3-Tabellen finden Sie unter <https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-tables-integrating-aws.html>.

Erstellen von Metriken aus Protokollereignissen mithilfe von Filtern

Sie können die Protokolldaten, die in CloudWatch Logs eingehen, durchsuchen und filtern, indem Sie einen oder mehrere Metrikfilter erstellen. Metrikfilter definieren die Begriffe und Muster, nach denen in Protokolldaten gesucht werden muss, wenn diese an CloudWatch Logs gesendet werden. CloudWatch Logs verwendet diese Metrikfilter, um Protokolldaten in numerische CloudWatch Messwerte umzuwandeln, die Sie grafisch darstellen oder einen Alarm auslösen können.

Wenn Sie eine Metrik aus einem Protokollfilter erstellen, können Sie der Metrik auch Dimensionen und eine Einheit zuweisen. Wenn Sie eine Einheit angeben, stellen Sie sicher, dass Sie beim Erstellen des Filters die richtige Einheit angeben. Wenn Sie die Einheit für den Filter später ändern, hat das keine Wirkung.

Wenn Sie Mitgliedskonten konfiguriert haben AWS Organizations und mit diesen arbeiten, können Sie die Protokollzentralisierung verwenden, um Protokolldaten von Quellkonten in einem zentralen Überwachungskonto zu sammeln.

Wenn Sie mit zentralisierten Protokollgruppen arbeiten, können Sie diese Systemfelddimensionen bei der Erstellung von Metrikfiltern verwenden.

- `@aws.account`- Diese Dimension stellt die AWS Konto-ID dar, von der das Protokollereignis ausgegangen ist.
- `@aws.region`- Diese Dimension stellt die AWS Region dar, in der das Protokollereignis generiert wurde.

Diese Dimensionen helfen bei der Identifizierung der Quelle der Protokolldaten und ermöglichen so eine detailliertere Filterung und Analyse von Metriken, die aus zentralisierten Protokollen stammen. Weitere Informationen finden Sie unter [Cross-account Regionsübergreifende Protokollzentralisierung](#).

Wenn eine Protokollgruppe mit einem Abonnement die Protokolltransformation verwendet, wird das Filtermuster auf die transformierten Versionen der Protokollereignisse angewendet. Weitere Informationen finden Sie unter [Logs während der Aufnahme transformieren](#).

 Note

Metrikfilter werden nur für Protokollgruppen der Standard-Protokollklasse unterstützt. Weitere Hinweise zu Protokollklassen finden Sie unter [Klassen protokollieren](#).

Sie können jede Art von CloudWatch Statistik verwenden, einschließlich Perzentilstatistiken, wenn Sie sich diese Metriken ansehen oder Alarme einstellen.

 Note

Perzentil-Statistiken werden für eine Metrik nur dann unterstützt, wenn keiner der Metrikwerte negativ ist. Wenn Sie Ihren Metrikfilter so einrichten, dass er negative Zahlen melden kann, stehen für diese Metrik keine Perzentilstatistiken zur Verfügung, wenn negative Zahlen als Werte angezeigt werden. Weitere Informationen finden Sie unter [Perzentile](#).

Filter können nicht rückwirkend auf Daten angewendet werden. Filter veröffentlichen nur die Metrikdatenpunkte für Ereignisse, die aufgetreten sind, nachdem der Filter erstellt wurde. Beim Testen eines Filtermusters zeigt die Vorschau der Filterergebnisse zu Überprüfungszwecken bis zu die ersten 50 übereinstimmenden Protokollzeilen an. Wenn der Zeitstempel der gefilterten Ergebnisse vor dem Zeitpunkt der Erstellung der Metrik liegt, werden keine Protokolle angezeigt.

Inhalt

- [Konzepte](#)
- [Filtermustersyntax für metrische Filter](#)
- [Erstellen von Metrikfiltern](#)
- [Auflisten von Metrikfiltern](#)
- [Löschen eines Metrikfilters](#)

Konzepte

Jeder Metrikfilter besteht aus den folgenden Schlüsselementen:

Standardwert

Der Wert, der dem Metrikfilter während eines Zeitraums gemeldet wird, in dem zwar Protokolle erfasst, aber keine passenden Protokolle gefunden werden. Wenn Sie diesen Wert auf 0 setzen, stellen Sie sicher, dass die Daten in jedem dieser Zeiträume gemeldet werden, und verhindern so „lückenhafte“ Metriken mit Zeiträumen ohne übereinstimmende Daten. Wenn während eines einminütigen Zeitraums keine Protokolle erfasst werden, wird kein Wert gemeldet.

Wenn Sie einer Metrik, die von einem Metrikfilter erstellt wurde, Dimensionen zuweisen, können Sie dieser Metrik keinen Standardwert zuweisen.

Dimensionen

Dimensionen sind die Schlüssel-Wert-Paare, die eine Metrik weiter definieren. Sie können der Metrik, die aus einem Metrikfilter erstellt wurde, Dimensionen zuweisen. Da Dimensionen Teil der eindeutigen Kennung für eine Metrik sind, erstellen Sie jedes Mal, wenn ein eindeutiges name/value Paar aus Ihren Logs extrahiert wird, eine neue Variante dieser Metrik.

Filtermuster

Eine symbolische Beschreibung, wie CloudWatch Logs die Daten in den einzelnen Protokollereignissen interpretieren sollte. Ein Protokolleintrag enthält z. B. möglicherweise Zeitstempel, IP-Adressen, Zeichenfolgen und so weiter. Mit dem Muster können Sie angeben, wonach in der Protokolldatei gesucht werden soll.

Metrikname

Der Name der CloudWatch Metrik, für die die überwachten Protokollinformationen veröffentlicht werden sollen. Sie können zum Beispiel mit einer Metrik namens veröffentlichen ErrorCount.

Namespace der Metrik

Der Ziel-Namespace der neuen CloudWatch Metrik.

Metrikwert

Der numerische Wert, der jedes Mal für die Metrik veröffentlicht werden soll, wenn ein übereinstimmendes Protokoll gefunden wird. Wenn Sie beispielsweise die Häufigkeit des Vorkommens eines bestimmten Begriffs wie "Fehler" zählen, lautet der Wert für jedes Vorkommen "1". Wenn Sie die übertragenen Bytes zählen, können Sie die Zahl um die tatsächliche Anzahl der Bytes, die im Protokollereignis gefunden wurden, erhöhen.

Filtermustersyntax für metrische Filter

Note

Wie sich Metrikfilter von CloudWatch Logs Insights-Abfragen unterscheiden
Metrikfilter unterscheiden sich von CloudWatch Logs Insights-Abfragen darin, dass einem Metrikfilter jedes Mal, wenn ein passendes Protokoll gefunden wird, ein bestimmter numerischer Wert hinzugefügt wird. Weitere Informationen finden Sie unter [Konfigurieren von Metrikwerten für einen Metrikfilter](#).

Informationen darüber, wie Sie Ihre Protokollgruppen mit der Abfragesprache von Amazon CloudWatch Logs Insights abfragen, finden Sie unter [CloudWatch Abfragesyntax in Logs Insights](#).

Beispiele für generische Filtermuster

Weitere Informationen zur generischen Filtermustersyntax für Metrikfilter sowie [Abonnementfilter](#) und [Filterprotokollereignisse](#) finden Sie unter [Filtermustersyntax für Metrikfilter, Abonnementfilter und Filterprotokollereignisse](#), unter anderem die folgenden:

- Unterstützte Syntax für reguläre Ausdrücke (Regex)
- Abgleichen von Begriffen in unstrukturierten Protokollereignissen
- Auffinden von Begriffen in JSON-Protokollereignissen
- Abgleichen von Begriffen in Leerzeichen-getrennten Protokollereignissen

Mit Metrikfiltern können Sie in CloudWatch Logs eingehende Protokolldaten suchen und filtern, metrische Beobachtungen aus den gefilterten Protokolldaten extrahieren und die Datenpunkte in eine CloudWatch Logs-Metrik umwandeln. Sie definieren die Begriffe und Muster, nach denen in den Protokolldaten gesucht werden soll, wenn diese an CloudWatch Logs gesendet werden. Metrikfilter sind Protokollgruppen zugewiesen, und alle einer Protokollgruppe zugewiesenen Filter werden auf ihre Protokollstreams angewendet.

Wenn ein Metrikfilter mit einem Begriff übereinstimmt, erhöht er die Anzahl der Metrik um einen bestimmten metrischen Wert. Sie können z. B. einen Metrikfilter erstellen, der zählt, wie oft das Wort ERROR (FEHLER) in Ihren Protokollereignissen vorkommt.

Sie können den Metriken Maßeinheiten und Dimensionen zuweisen. Wenn Sie beispielsweise einen Metrikfilter erstellen, der zählt, wie oft das Wort ERROR (FEHLER) in Ihren Protokollereignissen vorkommt, können Sie eine Dimension angeben, die `Errorcode` genannt wird, um die Gesamtzahl

der Protokollereignisse anzuzeigen, die das Wort ERROR (FEHLER) enthalten, und die Daten nach gemeldeten Fehlercodes filtern.

 Tip

Wenn Sie einer Metrik eine Maßeinheit zuordnen, achten Sie darauf, dass Sie die richtige Einheit angeben. Wenn Sie die Einheit später ändern, wird Ihre Änderung möglicherweise nicht wirksam. Die vollständige Liste der CloudWatch unterstützten Einheiten finden Sie [MetricDatum](#) in der Amazon CloudWatch API-Referenz.

Themen

- [Konfigurieren von Metrikwerten für einen Metrikfilter](#)
- [Veröffentlichung von Dimensionen mit Metriken aus Werten in JSON- oder Leerzeichen-getrennten Protokollereignissen](#)
- [Verwendung von Werten in Protokollereignissen zum Erhöhen des Metrikwerts](#)

Konfigurieren von Metrikwerten für einen Metrikfilter

Wenn Sie einen Metrikfilter erstellen, definieren Sie Ihr Filtermuster und geben den Wert der Metrik und den Standardwert an. Sie können Metrikwerte für Zahlen, benannte Bezeichner oder numerische Bezeichner festlegen. Wenn Sie keinen Standardwert angeben, CloudWatch werden keine Daten gemeldet, wenn Ihr Metrikfilter keine Übereinstimmung findet. Es wird empfohlen, einen Standardwert anzugeben, auch wenn der Wert 0 ist. Wenn Sie einen Standardwert festlegen, können Daten genauer CloudWatch gemeldet werden, und es wird CloudWatch verhindert, dass lückenhafte Messwerte aggregiert werden. CloudWatch aggregiert und meldet Metrikwerte jede Minute.

Wenn der Metrikfilter eine Übereinstimmung in Ihren Protokollereignissen findet, erhöht er die Metrikanzahl um den Wert der Metrik. Wenn Ihr Metrikfilter keine Übereinstimmung findet, wird der Standardwert der Metrik CloudWatch gemeldet. Beispiel: Ihre Protokollgruppe veröffentlicht jede Minute zwei Datensätze, der Metrikwert ist 1 und der Standardwert ist 0. Wenn der Metrikfilter in beiden Protokolleinträgen innerhalb der ersten Minute Übereinstimmungen findet, ist der Metrikwert für diese Minute 2. Wenn der Metrikfilter in beiden Datensätzen keine Übereinstimmungen während der zweiten Minute findet, ist der Standardwert für diese Minute 0. Wenn Sie den von den Metrikfiltern generierten Metriken Dimensionen zuweisen, können Sie keine Standardwerte für diese Metriken angeben.

Sie können auch einen Metrikfilter einrichten, um eine Metrik mit einem Wert zu erhöhen, der aus einem Protokollereignis extrahiert wurde, anstatt mit einem statischen Wert. Weitere Informationen finden Sie unter [Verwendung von Werten in Protokollereignissen zum Erhöhen des Metrikwerts](#).

Veröffentlichung von Dimensionen mit Metriken aus Werten in JSON- oder Leerzeichen-getrennten Protokollereignissen

Sie können die CloudWatch Konsole oder AWS CLI verwenden, um Metrikfilter zu erstellen, die Dimensionen mit Metriken veröffentlichen, die JSON und durch Leerzeichen getrennte Protokollereignisse generieren. Dimensionen sind name/value Wertepaare und nur für JSON- und durch Leerzeichen getrennte Filtermuster verfügbar. Sie können JSON- und Leerzeichen-getrennte Metrikfilter mit bis zu drei Dimensionen erstellen. Weitere Informationen zu Dimensionen und der Zuordnung von Dimensionen zu Metriken finden Sie in den folgenden Abschnitten:

- [Abmessungen](#) im CloudWatch Amazon-Benutzerhandbuch
- [Beispiel: Extrahieren von Feldern aus einem Apache-Protokoll und Zuweisen von Dimensionen](#) im Amazon CloudWatch Logs-Benutzerhandbuch

Important

Dimensionen enthalten Werte, die die gleichen Kosten wie benutzerdefinierte Metriken erzeugen. Um unerwartete Kosten zu vermeiden, sollten Sie keine Felder mit hoher Kardinalität, wie z. B. `IPAddress` oder `requestID`, als Dimensionen angeben.

Wenn Sie Metriken aus Protokollereignissen extrahieren, werden Ihnen benutzerdefinierte Metriken in Rechnung gestellt. Um zu verhindern, dass Sie versehentlich hohe Gebühren erheben, deaktiviert Amazon möglicherweise Ihren Metrikfilter, wenn er über einen bestimmten Zeitraum 1000 verschiedene name/value Paare für bestimmte Dimensionen generiert.

Sie können Rechnungsalarme erstellen, die Sie über Ihre geschätzten Gebühren informieren. Weitere Informationen finden Sie unter [Erstellen eines Rechnungsalarms zur Überwachung Ihrer geschätzten AWS -Gebühren](#).

Dimensionen mit Metriken aus JSON-Protokollereignissen veröffentlichen

Die folgenden Beispiele enthalten Code-Snippets, die beschreiben, wie in einem JSON-Metrikfilter Dimensionen angegeben werden.

Example: JSON log event

```
{
  "eventType": "UpdateTrail",
  "sourceIPAddress": "111.111.111.111",
  "arrayKey": [
    "value",
    "another value"
  ],
  "objectList": [
    {"name": "a",
     "id": 1
    },
    {"name": "b",
     "id": 2
    }
  ]
}
```

Note

Wenn Sie die Beispielmetrikfilter mit dem Beispiel-JSON-Protokollereignis testen, müssen Sie das Beispiel-JSON-Protokoll in einer einzigen Zeile eingeben.

Example: Metric filter

Der Metrikfilter erhöht die Metrik immer dann, wenn ein JSON-Protokollereignis die Eigenschaften `eventType` und `"sourceIPAddress"` enthält.

```
{ $.eventType = "*" && $.sourceIPAddress != 123.123.* }
```

Wenn Sie einen JSON-Metrikfilter erstellen, können Sie jede der Eigenschaften im Metrikfilter als Dimension angeben. Um `eventType` als Dimension festzulegen, verwenden Sie zum Beispiel Folgendes:

```
"eventType" : $.eventType
```

Die Beispielmetrik enthält eine Dimension mit dem Namen "eventType", und der Wert der Dimension im Beispiel-Protokollereignis ist "UpdateTrail".

Publizieren von Dimensionen mit Metriken aus durch Leerzeichen getrennten Protokollereignissen

Die folgenden Beispiele enthalten Code-Snippets, die beschreiben, wie in einem Leerzeichen-getrennten Metrikfilter Dimensionen angegeben werden.

Example: Space-delimited log event

```
127.0.0.1 Prod frank [10/Oct/2000:13:25:15 -0700] "GET /index.html HTTP/1.0" 404
1534
```

Example: Metric filter

```
[ip, server, username, timestamp, request, status_code, bytes > 1000]
```

Der Metrikfilter erhöht die Metrik, wenn ein Leerzeichen-getrenntes Protokollereignis eines der im Filter angegebenen Felder enthält. Zum Beispiel findet der Metrikfilter die folgenden Felder und Werte im Leerzeichen-getrennten Beispiel-Protokollereignis.

```
{
  "$bytes": "1534",
  "$status_code": "404",

  "$request": "GET /index.html HTTP/1.0",
  "$timestamp": "10/Oct/2000:13:25:15 -0700",
  "$username": "frank",
  "$server": "Prod",
  "$ip": "127.0.0.1"
```

```
}
```

Wenn Sie einen Leerzeichen-getrennten Metrikfilter erstellen, können Sie jedes der Felder im Metrikfilter als Dimension angeben. Um `server` als Dimension festzulegen, verwenden Sie zum Beispiel Folgendes:

```
"server" : $server
```

Der Beispiel-Metrikfilter hat eine Dimension mit dem Namen `server`, und der Wert der Dimension im Beispiel-Protokollereignis ist `"Prod"`.

Example: Match terms with AND (&&) and OR (||)

Sie können die logischen Operatoren AND („&&“) und OR („||“) verwenden, um Leerzeichen-getrennte Metrikfilter zu erstellen, die Bedingungen enthalten. Der folgende Metrikfilter gibt Protokollereignisse zurück, bei denen der erste Begriff in den Ereignissen `ERROR` (FEHLER) oder eine beliebige `WARN`-Zeichenfolge ist.

```
[w1=ERROR || w1=%WARN%, w2]
```

Verwendung von Werten in Protokollereignissen zum Erhöhen des Metrikwerts

Sie können Metrikfilter erstellen, die numerische Werte aus Ihren Protokollereignissen veröffentlichen. Die Vorgehensweise in diesem Abschnitt zeigt anhand des folgenden Beispiels für einen Metrikfilter, wie Sie einen numerischen Wert in einem JSON-Protokollereignis in einer Metrik veröffentlichen können.

```
{ $.latency = * } metricValue: $.latency
```

Einen Metrikfilter erstellen, der einen Wert in einem Protokollereignis veröffentlicht

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Log groups (Protokollgruppen) aus.

3. Wählen oder erstellen Sie eine Protokollgruppe.

Informationen zum Erstellen einer Protokollgruppe finden Sie unter [Erstellen einer Protokollgruppe in CloudWatch Logs](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

4. Wählen Sie Aktionen und dann Metrikfilter erstellen.

5. Geben Sie unter Filter Pattern (Filtermuster) den Wert `{ $.latency = * }` ein und wählen Sie dann Next (Weiter).

6. Geben Sie für Metric Name (Metrikname) myMetric ein.

7. Geben Sie für Metrikwert `$.latency` ein.

8. Geben Sie für Default Value (Standardwert) 0 ein und wählen Sie dann Next (Weiter).

Es wird empfohlen, einen Standardwert anzugeben, auch wenn der Wert 0 ist. Wenn Sie einen Standardwert festlegen, können Daten genauer CloudWatch gemeldet werden, und es wird CloudWatch verhindert, dass lückenhafte Messwerte aggregiert werden. CloudWatch aggregiert und meldet Metrikwerte jede Minute.

9. Wählen Sie Metrikfilter erstellen aus.

Der Beispiel-Metrikfilter stimmt mit dem Begriff "latency" im Beispiel-JSON-Protokollereignis überein und veröffentlicht einen numerischen Wert von 50 für die Metrik myMetric.

```
{
  "latency": 50,
  "requestType": "GET"
}
```

Erstellen von Metrikfiltern

Die folgenden Schritte und Beispiele zeigen, wie Sie Metrikfilter erstellen.

Beispiele

- [Erstellen eines Metrikfilters für eine Protokollgruppe](#)
- [Beispiel: Zählen von Protokollereignissen](#)
- [Beispiel: Zählen des Vorkommens eines Begriffs](#)
- [Beispiel: Zählen von HTTP-404-Codes](#)

- [Beispiel: Zählen von HTTP-4xx-Codes](#)
- [Beispiel: Extrahieren von Feldern aus einem Apache-Protokoll und Zuweisen von Dimensionen](#)

Erstellen eines Metrikfilters für eine Protokollgruppe

Gehen Sie wie folgt vor, um einen Metrikfilter für eine Protokollgruppe zu erstellen. Die Metrik wird erst angezeigt, wenn einige Datenpunkte dafür vorliegen.

Um einen Metrikfilter mit der CloudWatch Konsole zu erstellen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Logs (Protokolle) und dann Log groups (Protokollgruppen) aus.
3. Wählen Sie den Namen der Protokollgruppe aus.
4. Wählen Sie Actions und dann Create metric filter (Metrikfilter erstellen).
5. Für Filter pattern (Filtermuster) geben Sie ein Filtermuster ein. Weitere Informationen finden Sie unter [Filtermustersyntax für Metrikfilter, Abonnementfilter, Filterprotokollereignisse und Live Tail](#).
6. (Optional) Wenn Sie zentralisierte Protokollgruppen verwenden, können Sie unter Filterauswahlkriterien Filter angeben, die auf dem Quellkonto (@aws . account), der Quellregion (@aws . region) oder beiden Bedingungen basieren.
7. (Optional) Um das Filtermuster zu testen, geben Sie unter Test Pattern (Muster testen) ein oder mehrere Protokollereignisse zum Testen des Musters ein. Jedes Protokollereignis muss in einer Zeile formatiert sein. Zeilenumbrüche werden verwendet, um Protokollereignisse in dem Kästchen Log event messages (Protokollereignisnachrichten) zu trennen.
8. Wählen Sie Next (Weiter) aus und geben Sie einen Namen für Ihren Metrikfilter ein.
9. Geben Sie unter Metrikdetails für Metrik-Namespace einen Namen für den CloudWatch Namespace ein, in dem die Metrik veröffentlicht wird. Wenn der Namespace noch nicht vorhanden ist, stellen Sie sicher, dass Create new (Neu erstellen) ausgewählt ist.
10. Unter Metric name (Metrikname) geben Sie einen Namen für die neue Metrik ein.
11. Geben Sie für Metric value (Metrikwert) 1 ein, wenn der Metrikfilter Vorkommen der Schlüsselwörter im Filter zählt. Dadurch wird die Metrik für jedes Protokollereignis, das eines der Schlüsselwörter enthält, um 1 erhöht.

Alternativ können Sie ein Token wie `$size` eingeben. Dadurch wird die Metrik für jedes Protokollereignis, das ein `size`-Feld enthält, um den Wert der Zahl in der `size` erhöht.

12. (Optional) Wählen Sie für Unit (Einheit) eine Einheit aus, die der Metrik zugewiesen werden soll. Wenn Sie keine Einheit angeben, wird None als Einheit eingestellt.
13. (Optional) Geben Sie die Namen und Token für bis zu drei Dimensionen für die Metrik ein. Wenn Sie Metriken, die von Metrikfiltern erstellt werden, Dimensionen zuweisen, können Sie diesen Metriken keine Standardwerte zuweisen.

 Note

Dimensionen werden nur in JSON- oder durch Leerzeichen getrennten Metrikfiltern unterstützt.

14. Wählen Sie Metrikfilter erstellen aus. Sie finden den Metrikfilter, den Sie über den Navigationsbereich erstellt haben. Wählen Sie Logs (Protokolle) und anschließend Log groups (Protokollgruppen) aus. Wählen Sie den Namen der Protokollgruppe aus, für die Sie den Metrikfilter erstellt haben, und wählen Sie dann den Tabulator Metric filters (Metrikfilter).

Beispiel: Zählen von Protokollereignissen

Die einfachste Art der Überwachung von Protokollereignissen ist die Zählung der Anzahl von Protokollereignissen. Dies ist sinnvoll, um alle Ereignisse zu zählen, um eine "Heartbeat"-Überwachung zu erstellen oder um lediglich die Erstellung von Metrikfiltern zu üben.

Im folgenden CLI-Beispiel MyAppAccessCount wird ein Metrikfilter namens auf die Protokollgruppe MyApp /access.log angewendet, um die Metrik EventCount im CloudWatch Namespace MyNamespace zu erstellen. Der Filter ist so konfiguriert, dass er dem Inhalt eines beliebigen Protokollereignisses entspricht und die Metrik um "1" erhöht wird.

Um einen Metrikfilter mit der CloudWatch Konsole zu erstellen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Protokollgruppen aus.
3. Wählen Sie den Namen einer Protokollgruppe aus.
4. Wählen Sie Actions, Metrikfilter erstellen aus.
5. Lassen Sie die Felder Filtermuster und Zu testende Protokolldaten auswählen leer.
6. Wählen Sie Weiter und geben Sie dann für Filtername **EventCount** ein.

7. Geben Sie unter Metric Details (Metrikdetails) für Metric Namespace (Metrik-Namespace) **MyNameSpace** ein.
8. Geben Sie für Metrikname den Wert **MyAppEventCount** ein.
9. Vergewissern Sie sich, dass Metrikwert „1“ lautet. Dadurch ist festgelegt, dass die Zählung für jedes Protokollereignis um 1 erhöht wird.
10. Geben Sie für Standardwert „0“ ein und wählen Sie dann Weiter. Die Angabe eines Standardwerts stellt sicher, dass auch in Zeiträumen, in denen keine Protokollereignisse auftreten, Daten gemeldet werden, und verhindert so lückenhafte Metriken, in denen manchmal gar keine Daten vorkommen.
11. Wählen Sie Metrikfilter erstellen aus.

Um einen metrischen Filter mit dem AWS CLI

Führen Sie über die Eingabeaufforderung den folgenden Befehl aus:

```
aws logs put-metric-filter \  
  --log-group-name MyApp/access.log \  
  --filter-name EventCount \  
  --filter-pattern " " \  
  --metric-transformations \  
  metricName=MyAppEventCount,metricNamespace=MyNameSpace,metricValue=1,defaultValue=0
```

Sie können diese neue Richtlinie durch Veröffentlichen beliebiger Ereignisdaten testen. Sie sollten Datenpunkte sehen, die in der Metrik veröffentlicht wurden MyAppAccessEventCount.

Um Ereignisdaten mit dem zu posten AWS CLI

Führen Sie über die Eingabeaufforderung den folgenden Befehl aus:

```
aws logs put-log-events \  
  --log-group-name MyApp/access.log --log-stream-name TestStream1 \  
  --log-events \  
    timestamp=1394793518000,message="Test event 1" \  
    timestamp=1394793518000,message="Test event 2" \  
    timestamp=1394793528000,message="This message also contains an Error"
```

Beispiel: Zählen des Vorkommens eines Begriffs

Protokollereignisse enthalten häufig wichtige Nachrichten (z. B. über den Erfolg oder Misserfolg von Operationen), die Sie zählen möchten. Beispielsweise kann ein Fehler auftreten und in einer Protokolldatei aufgezeichnet werden, wenn eine bestimmte Operation fehlschlägt. Sie sollten diese Einträge überwachen, um sich einen Überblick über die Entwicklung Ihrer Fehler zu verschaffen.

In dem unten stehenden Beispiel wird ein Metrikfilter erstellt, mit dem der Begriff "Fehler" überwacht wird. Die Richtlinie wurde erstellt und der Protokollgruppe MyApp/message.log hinzugefügt. CloudWatch Logs veröffentlicht einen Datenpunkt für die CloudWatch benutzerdefinierte Metrik ErrorCount im Namespace MyApp/message.log mit dem Wert „1“ für jedes Ereignis, das Fehler enthält. Wenn kein Ereignis das Wort "Error" enthält, werden der Wert 0 veröffentlicht. Achten Sie bei der grafischen Darstellung dieser Daten in der CloudWatch Konsole darauf, die Summenstatistik zu verwenden.

Nachdem Sie einen Metrikfilter erstellt haben, können Sie die Metrik in der CloudWatch Konsole anzeigen. Wenn Sie die anzuzeigende Metrik auswählen, wählen Sie den Metrik-Namespace aus, der mit dem Namen der Protokollgruppe übereinstimmt. Weitere Informationen finden Sie unter [Anzeigen der verfügbaren Metriken](#).

Um einen Metrikfilter mit der CloudWatch Konsole zu erstellen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Protokollgruppen aus.
3. Wählen Sie den Namen der Protokollgruppe aus.
4. Wählen Sie Aktionen, Metrikfilter erstellen aus.
5. Geben Sie für Filtermuster **Error** ein.

Note

Alle Einträge in Filter Pattern berücksichtigen Groß- und Kleinschreibung.

6. (Optional) Um das Filtermuster zu testen, geben Sie unter Test Pattern (Muster testen) ein oder mehrere Protokollereignisse ein, die zum Testen des Musters verwendet werden sollen. Jedes Protokollereignis muss innerhalb einer einzelnen Zeile liegen, da Zeilenumbrüche verwendet werden, um Protokollereignisse im Anzeigebereich Log event messages (Ereignismeldungen protokollieren) zu trennen.

7. Wählen Sie Weiter und geben Sie dann auf der Seite Metrik zuweisen für Filtername **MyAppErrorCount** ein.
8. Geben Sie unter Metric Details (Metrikdetails) für Metric Namespace (Metrik-Namespace) MyNamespace ein.
9. Geben Sie für Metrikname den Wert ErrorCount ein.
10. Vergewissern Sie sich, dass Metrikwert „1“ lautet. Dadurch ist festgelegt, dass die Zählung für jedes Protokollereignis, das „Error“ enthält, um 1 erhöht wird.
11. Geben Sie für Standardwert „0“ ein und wählen Sie dann Weiter.
12. Wählen Sie Metrikfilter erstellen aus.

Um einen metrischen Filter mit dem AWS CLI

Führen Sie über die Eingabeaufforderung den folgenden Befehl aus:

```
aws logs put-metric-filter \  
  --log-group-name MyApp/message.log \  
  --filter-name MyAppErrorCount \  
  --filter-pattern 'Error' \  
  --metric-transformations \  
    metricName=ErrorCount,metricNamespace=MyNamespace,metricValue=1,defaultValue=0
```

Sie können diese neue Richtlinie testen, indem Sie Ereignisse mit dem Begriff "Fehler" in der Nachricht veröffentlichen.

Um Ereignisse zu posten mit dem AWS CLI

Führen Sie an einer Eingabeaufforderung den folgenden -Befehl aus. Beachten Sie, dass in Mustern Groß- und Kleinschreibung berücksichtigt wird.

```
aws logs put-log-events \  
  --log-group-name MyApp/access.log --log-stream-name TestStream1 \  
  --log-events \  
    timestamp=1394793518000,message="This message contains an Error" \  
    timestamp=1394793528000,message="This message also contains an Error"
```

Beispiel: Zählen von HTTP-404-Codes

Mithilfe von CloudWatch Logs können Sie überwachen, wie oft Ihre Apache-Server eine HTTP 404-Antwort zurückgeben. Dabei handelt es sich um den Antwortcode für die Seite, die nicht gefunden

wurde. Sie sollten diese Zahl überwachen, um zu verstehen, wie oft die Besucher Ihrer Website die gesuchte Ressource nicht finden. Gehen Sie davon aus, dass die Protokolldatensätze so strukturiert sind, dass sie folgende Informationen für jedes Protokollereignis (Besuch der Website) enthalten:

- IP-Adresse des Anforderers
- RFC 1413-Identität
- Username
- Zeitstempel
- Anforderungsmethode mit angeforderter Ressource und Protokoll
- Anzufordernder HTTP-Antwortcode
- Bei der Anforderung übertragene Bytes.

Ein Beispiel könnte folgendermaßen aussehen:

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 404 2326
```

Sie können eine Regel festlegen, die versucht, Ereignisse in der Struktur der HTTP 404-Fehler zu finden. Dies ist in folgendem Beispiel dargestellt:

Um einen Metrikfilter mit der CloudWatch Konsole zu erstellen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Protokollgruppen aus.
3. Wählen Sie Actions, Metrikfilter erstellen aus.
4. Geben Sie für Filtermuster **[IP, UserInfo, User, Timestamp, RequestInfo, StatusCode=404, Bytes]** ein.
5. (Optional) Um das Filtermuster zu testen, geben Sie unter Test Pattern (Muster testen) ein oder mehrere Protokollereignisse ein, die zum Testen des Musters verwendet werden sollen. Jedes Protokollereignis muss innerhalb einer einzelnen Zeile liegen, da Zeilenumbrüche verwendet werden, um Protokollereignisse im Anzeigebereich Log event messages (Ereignismeldungen protokollieren) zu trennen.
6. Wählen Sie Weiter und geben Sie dann als Filtername HTTP404 Fehler ein.
7. Geben Sie unter Metric Details (Metrikdetails) für Metric Namespace (Metrik-Namespace) **MyNameSpace** ein.

8. Geben Sie für Metrikname **ApacheNotFoundErrorCode** ein.
9. Vergewissern Sie sich, dass Metrikwert „1“ lautet. Dadurch ist festgelegt, dass die Zählung für jedes Protokollereignis „404 Error“ um 1 erhöht wird.
10. Geben Sie für Standardwert „0“ ein und wählen Sie dann Weiter.
11. Wählen Sie Metrikfilter erstellen aus.

Um einen Metrikfilter zu erstellen, verwenden Sie AWS CLI

Führen Sie über die Eingabeaufforderung den folgenden Befehl aus:

```
aws logs put-metric-filter \
  --log-group-name MyApp/access.log \
  --filter-name HTTP404Errors \
  --filter-pattern '[ip, id, user, timestamp, request, status_code=404, size]' \
  --metric-transformations \
    metricName=ApacheNotFoundErrorCode,metricNamespace=MyNamespace,metricValue=1
```

In diesem Beispiel werden Literalzeichen wie die linke und rechte eckige Klammern, Anführungszeichen und die Zeichenfolge 404 verwendet. Das Muster muss mit der gesamten Protokollereignisnachricht für das zu überwachende Protokollereignis übereinstimmen.

Sie können überprüfen, ob der Metrikfilter erstellt wurde, indem Sie den Befehl `describe-metric-filters` ausführen. Die Ausgabe sollte in etwa wie folgt aussehen:

```
aws logs describe-metric-filters --log-group-name MyApp/access.log

{
  "metricFilters": [
    {
      "filterName": "HTTP404Errors",
      "metricTransformations": [
        {
          "metricValue": "1",
          "metricNamespace": "MyNamespace",
          "metricName": "ApacheNotFoundErrorCode"
        }
      ],
      "creationTime": 1399277571078,
      "filterPattern": "[ip, id, user, timestamp, request, status_code=404, size]"
    }
  ]
}
```



```
}  
]  
}
```

Jetzt können Sie einige Ereignisse manuell veröffentlichen:

```
aws logs put-log-events \  
--log-group-name MyApp/access.log --log-stream-name hostname \  
--log-events \  
timestamp=1394793518000,message="127.0.0.1 - bob [10/Oct/2000:13:55:36 -0700] \"GET /  
apache_pb.gif HTTP/1.0\" 404 2326" \  
timestamp=1394793528000,message="127.0.0.1 - bob [10/Oct/2000:13:55:36 -0700] \"GET /  
apache_pb2.gif HTTP/1.0\" 200 2326"
```

Kurz nachdem Sie diese Beispielprotokollereignisse gespeichert haben, können Sie die Metrik abrufen, die in der CloudWatch Konsole als benannt ist `ApacheNotFoundErrorCount`.

Beispiel: Zählen von HTTP-4xx-Codes

Wie im vorherigen Beispiel sollten Sie die Zugriffsprotokolle für Webservices sowie die Ebenen des HTTP-Antwortcodes überwachen. Sie sollten beispielsweise alle Fehler auf HTTP-400-Ebene überwachen. Sie sollten jedoch einen neuen Metrikfilter für jeden Rückgabecode angeben.

Im folgenden Beispiel wird gezeigt, wie Sie eine Metrik erstellen, die alle Antworten auf der Ebene des HTTP 400-Codes aus einem Zugriffsprotokoll enthält, und dazu das Format des Apache-Zugriffsprotokolls aus dem Beispiel [Beispiel: Zählen von HTTP-404-Codes](#) verwenden.

Um einen Metrikfilter mit der CloudWatch Konsole zu erstellen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Protokollgruppen aus.
3. Wählen Sie den Namen der Protokollgruppe für den Apache-Server aus.
4. Wählen Sie Actions, Metrikfilter erstellen aus.
5. Geben Sie für Filter pattern (Filtermuster) **[ip, id, user, timestamp, request, status_code=4*, size]** ein.
6. (Optional) Um das Filtermuster zu testen, geben Sie unter Test Pattern (Muster testen) ein oder mehrere Protokollereignisse ein, die zum Testen des Musters verwendet werden sollen. Jedes Protokollereignis muss innerhalb einer einzelnen Zeile liegen, da Zeilenumbrüche verwendet

werden, um Protokollereignisse im Anzeigebereich Log event messages (Ereignismeldungen protokollieren) zu trennen.

7. Wählen Sie Next (Weiter) und geben Sie dann für Filter name (Filtername) **HTTP4xxErrors** ein.
8. Geben Sie unter Metric details (Metrikdetails) für Metric Namespace (Metrik-Namespace) **MyNameSpace** ein.
9. Geben Sie als Metriknamen HTTP4xx Errors ein.
10. Geben Sie für Metric value (Metrikwert) 1 ein. Dadurch ist festgelegt, dass die Zählung für jedes Protokollereignis, das einen „4xx“-Fehler enthält, um 1 erhöht wird.
11. Geben Sie für Default value (Standardwert) 0 ein und wählen Sie dann Next (Weiter).
12. Wählen Sie Metrikfilter erstellen aus.

Um einen Metrikfilter zu erstellen, verwenden Sie AWS CLI

Führen Sie über die Eingabeaufforderung den folgenden Befehl aus:

```
aws logs put-metric-filter \  
  --log-group-name MyApp/access.log \  
  --filter-name HTTP4xxErrors \  
  --filter-pattern '[ip, id, user, timestamp, request, status_code=4*, size]' \  
  --metric-transformations \  
    metricName=HTTP4xxErrors,metricNamespace=MyNameSpace,metricValue=1,defaultValue=0
```

Sie können die folgenden Daten in put-Ereignisaufrufen verwenden, um diese Regel zu testen. Wenn Sie die Überwachungsregel im vorherigen Beispiel nicht entfernt haben, werden zwei verschiedene Metriken generiert.

```
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287  
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287  
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /~test/ HTTP/1.1" 200 3  
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308  
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308  
127.0.0.1 - - [24/Sep/2013:11:51:34 -0700] "GET /~test/index.html HTTP/1.1" 200 3
```

Beispiel: Extrahieren von Feldern aus einem Apache-Protokoll und Zuweisen von Dimensionen

Manchmal ist es hilfreich, Werte aus einzelnen Protokollereignissen für Metrikwerte zu verwenden statt sie zu zählen. In diesem Beispiel wird gezeigt, wie Sie eine Extraktionsregel zum Erstellen einer Metrik erstellen können, mit der die von einem Apache-Webserver übertragenen Bytes gemessen werden.

In diesem Beispiel wird auch gezeigt, wie Sie der Metrik, die Sie erstellen, Dimensionen zuweisen.

Um einen Metrikfilter mit der CloudWatch Konsole zu erstellen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Protokollgruppen aus.
3. Wählen Sie den Namen der Protokollgruppe für den Apache-Server aus.
4. Wählen Sie Actions, Metrikfilter erstellen aus.
5. Geben Sie für Filter pattern (Filtermuster) **[ip, id, user, timestamp, request, status_code, size]** ein.
6. (Optional) Um das Filtermuster zu testen, geben Sie unter Test Pattern (Muster testen) ein oder mehrere Protokollereignisse ein, die zum Testen des Musters verwendet werden sollen. Jedes Protokollereignis muss innerhalb einer einzelnen Zeile liegen, da Zeilenumbrüche verwendet werden, um Protokollereignisse im Anzeigebereich Log event messages (Ereignismeldungen protokollieren) zu trennen.
7. Wählen Sie Next (Weiter) und geben Sie dann für Filter name (Filtername) **size** ein.
8. Geben Sie unter Metric details (Metrikdetails) für Metric Namespace (Metrik-Namespace) **MyNameSpace** ein. Da es sich um einen neuen Namespace handelt, stellen Sie sicher, dass Create new (Neu erstellen) ausgewählt ist.
9. Geben Sie für Metric name (Metrikname) **BytesTransferred** ein
10. Geben Sie für Metric value (Metrikwert) **\$size** ein.
11. Wählen Sie für Unit (Einheit) Byte aus.
12. Geben Sie unter Dimension Name die Zeichenfolge **IP** ein.
13. Geben Sie für Dimension Value (Dimensionswert) **\$ip** ein und wählen Sie dann Next (Weiter).
14. Wählen Sie Metrikfilter erstellen aus.

Um diesen Metrikfilter mit dem AWS CLI

Führen Sie über die Eingabeaufforderung den folgenden Befehl aus.

```
aws logs put-metric-filter \
--log-group-name MyApp/access.log \
--filter-name BytesTransferred \
--filter-pattern '[ip, id, user, timestamp, request, status_code, size]' \
--metric-transformations \
metricName=BytesTransferred,metricNamespace=MyNamespace,metricValue='$size'
```

```
aws logs put-metric-filter \
--log-group-name MyApp/access.log \
--filter-name BytesTransferred \
--filter-pattern '[ip, id, user, timestamp, request, status_code, size]' \
--metric-transformations \
metricName=BytesTransferred,metricNamespace=MyNamespace,metricValue='$size',unit=Bytes,dimension1=$ip}}'
```

Note

Verwenden Sie in diesem Befehl dieses Format, um mehrere Dimensionen anzugeben.

```
aws logs put-metric-filter \
--log-group-name my-log-group-name \
--filter-name my-filter-name \
--filter-pattern 'my-filter-pattern' \
--metric-transformations \
metricName=my-metric-name,metricNamespace=my-metric-namespace,metricValue=my-token,unit=unit,dimensions='{dimension1=$dim,dimension2=$dim2,dim3=$dim3}'
```

Sie können die folgenden Daten in put-log-event Aufrufen verwenden, um diese Regel zu testen. Damit werden zwei unterschiedliche Metriken erstellt, wenn Sie die Überwachungsregel im vorherigen Beispiel nicht entfernt haben.

```
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /~test/ HTTP/1.1" 200 3
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
```

```
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:51:34 -0700] "GET /~test/index.html HTTP/1.1" 200 3
```

Auflisten von Metrikfiltern

Sie können alle Metrikfilter in einer Protokollgruppe auflisten.

Um Metrikfilter mithilfe der CloudWatch Konsole aufzulisten

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Protokollgruppen aus.
3. Wählen Sie im Inhaltsbereich in der Liste der Protokollgruppen in der Spalte Metric Filters die Anzahl der Filter aus.

Auf dem Bildschirm Log Groups > Filters for werden alle Metrikfilter für die Protokollgruppe aufgelistet.

Um metrische Filter aufzulisten, verwenden Sie AWS CLI

Führen Sie über die Eingabeaufforderung den folgenden Befehl aus:

```
aws logs describe-metric-filters --log-group-name MyApp/access.log
```

Das Folgende ist eine Beispielausgabe:

```
{
  "metricFilters": [
    {
      "filterName": "HTTP404Errors",
      "metricTransformations": [
        {
          "metricValue": "1",
          "metricNamespace": "MyNamespace",
          "metricName": "ApacheNotFoundErrorCode"
        }
      ],
      "creationTime": 1399277571078,
      "filterPattern": "[ip, id, user, timestamp, request, status_code=404,
size]"
    }
  ]
}
```

```
]
}
```

Löschen eines Metrikfilters

Eine Richtlinie wird anhand des Namens und der entsprechenden Protokollgruppe identifiziert.

Um einen Metrikfilter mit der CloudWatch Konsole zu löschen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Protokollgruppen aus.
3. Wählen Sie im Inhaltsbereich in der Spalte Metric Filter (Metrikfilter) die Anzahl der Metrikfilter für die Protokollgruppe aus.
4. Wählen Sie unter Metric Filters (Metrikfilter) das Kontrollkästchen rechts neben dem Namen des Filters aus, den Sie löschen möchten. Wählen Sie dann Löschen aus.
5. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Delete (Löschen).

Um einen metrischen Filter mit dem zu löschen AWS CLI

Führen Sie über die Eingabeaufforderung den folgenden Befehl aus:

```
aws logs delete-metric-filter --log-group-name MyApp/access.log \  
--filter-name MyFilterName
```

Echtzeitverarbeitung von Protokolldaten mit Abonnements

Sie können Abonnements verwenden, um Zugriff auf einen Echtzeit-Feed mit Protokollereignissen aus CloudWatch Logs zu erhalten und ihn an andere Services wie einen Amazon Kinesis Kinesis-Stream, einen Amazon Data Firehose-Stream oder AWS Lambda zur benutzerdefinierten Verarbeitung, Analyse oder zum Laden auf andere Systeme liefern zu lassen. Wenn Protokollereignisse an den empfangenden Service gesendet werden, werden sie base64-kodiert und im GZIP-Format komprimiert.

Sie können die CloudWatch Logs-Zentralisierung auch verwenden, um Protokolldaten aus mehreren Konten und Regionen an einem zentralen Ort zu replizieren. Weitere Informationen finden Sie unter [Cross-account Regionsübergreifende Protokollzentralisierung](#).

Um mit dem Abonnieren von Protokollereignissen zu beginnen, erstellen Sie die empfangende Ressource, z. B. einen Amazon Kinesis Data Streams Streams-Stream, an den die Ereignisse übermittelt werden. Ein Abonnementfilter definiert das Filtermuster, das verwendet werden soll, um zu filtern, welche Protokollereignisse an Ihre AWS Ressource übermittelt werden, sowie Informationen darüber, wohin passende Protokollereignisse gesendet werden sollen. Protokollereignisse werden kurz nach der Aufnahme, in der Regel in weniger als drei Minuten, an die empfangende Ressource gesendet.

Note

Wenn eine Protokollgruppe mit einem Abonnement die Protokolltransformation verwendet, wird das Filtermuster mit den transformierten Versionen der Protokollereignisse verglichen. Weitere Informationen finden Sie unter [Logs während der Aufnahme transformieren](#).

Sie können Abonnements auf Konto- und Protokollgruppenebene erstellen. Jedes Konto kann einen Abonnementfilter auf Kontoebene pro Region haben. Jeder Protokollgruppe können bis zu zwei Abonnementfilter zugeordnet sein.

Note

Wenn der Zieldienst einen Fehler zurückgibt, der wiederholt werden kann, z. B. eine Drosselungs Ausnahme oder eine Dienstausschne, die erneut versucht werden kann (z. B. HTTP 5xx), wiederholt CloudWatch Logs die Zustellung für bis zu 24 Stunden.

CloudWatch Logs versucht nicht, erneut zuzustellen, wenn es sich bei dem Fehler um einen Fehler handelt, der nicht wiederholt werden kann, wie z. B. oder. `AccessDeniedException` `ResourceNotFoundException` In diesen Fällen ist der Abonnementfilter für bis zu 10 Minuten deaktiviert, und Logs versucht dann erneut, CloudWatch Protokolle an das Ziel zu senden. Während dieses deaktivierten Zeitraums werden Protokolle übersprungen.

CloudWatch Logs erstellt auch CloudWatch Messwerte über die Weiterleitung von Protokollereignissen an Abonnements. Weitere Informationen finden Sie unter [Überwachung mit CloudWatch Metriken](#).

Sie können auch ein CloudWatch Logs-Abonnement verwenden, um Protokolldaten nahezu in Echtzeit in einen Amazon OpenSearch Service-Cluster zu streamen. Weitere Informationen finden Sie unter [Streaming CloudWatch protokolliert Daten an Amazon OpenSearch Service](#).

Abonnements werden nur für Protokollgruppen der Standard-Protokollklasse unterstützt. Weitere Hinweise zu Protokollklassen finden Sie unter [Klassen protokollieren](#).

 Note

Abonnementfilter können Ereignisse stapelweise protokollieren, um die Übertragung zu optimieren und die Anzahl der Anrufe an das Ziel zu reduzieren. Die Batchverarbeitung ist nicht garantiert, wird aber nach Möglichkeit verwendet.

Für die Batch-Verarbeitung und Analyse von Protokolldaten nach einem Zeitplan sollten Sie die Verwendung von [Automatisieren der Protokollanalyse mit geplanten Abfragen](#). Geplante Abfragen führen CloudWatch Logs Insights-Abfragen automatisch aus und liefern Ergebnisse an Ziele wie Amazon S3 S3-Buckets oder Amazon EventBridge Event Buses.

 Note

Abonnementfilter sorgen dafür, dass Ereignisse mindestens einmal zugestellt werden, wobei gelegentlich doppelte Ereignisse auftreten können.

Inhalt

- [Konzepte](#)

- [Abonnementfilter auf Gruppenebene protokollieren](#)
- [Abonnementfilter auf Kontoebene](#)
- [Kontoübergreifende, regionsübergreifende Abonnements](#)
- [Confused-Deputy-Prävention](#)
- [Verhinderung der Rekursion von Protokollen](#)

Konzepte

Jeder Abonnementfilter besteht aus den folgenden Schlüsselementen:

Filtermuster

Eine symbolische Beschreibung, wie CloudWatch Logs die Daten in den einzelnen Protokollereignissen interpretieren soll, zusammen mit Filterausdrücken, die einschränken, was an die AWS Zielressource gesendet wird. Weitere Informationen über die Syntax von Filtermustern finden Sie unter [Filtermustersyntax für Metrikfilter, Abonnementfilter, Filterprotokollereignisse und Live Tail](#).

Ziel-ARN

Der Amazon-Ressourcenname (ARN) des Amazon Kinesis Data Streams-Streams, Firehose-Streams oder der Lambda-Funktion, die Sie als Ziel des Abonnement-Feeds verwenden möchten.

Rollen-ARN

Eine IAM-Rolle, die CloudWatch Logs die erforderlichen Berechtigungen erteilt, um Daten an das gewählte Ziel zu übertragen. Diese Rolle ist für Lambda-Ziele nicht erforderlich, da CloudWatch Logs die erforderlichen Berechtigungen aus den Zugriffskontrolleinstellungen der Lambda-Funktion selbst abrufen können.

Verteilung

Die Methode zur Verteilung von Protokolldaten an das Ziel, wenn es sich bei dem Ziel um einen Datenstrom in Amazon Kinesis Data Streams handelt. Standardmäßig werden Daten nach Protokoll-Stream gruppiert. Für eine gleichmäßige Verteilung können Sie Protokolldaten zufällig gruppieren.

Bei Protokollabonnements auf Gruppenebene ist das folgende Schlüsselement ebenfalls enthalten:

Name der Protokollgruppe

Die Protokollgruppe, die mit dem Abonnementfilter verknüpft ist. Alle Protokollereignisse, die in diese Protokollgruppe hochgeladen werden, unterliegen dem Abonnementfilter, und diejenigen, die dem Filter entsprechen, werden an den Zielservice gesendet, der die übereinstimmenden Protokollereignisse empfängt.

Bei Abonnements auf Kontoebene ist das folgende Schlüsselement ebenfalls enthalten:

Auswahlkriterien

Die Kriterien, anhand derer ausgewählt wurde, auf welche Protokollgruppen der Abonnementfilter auf Kontoebene angewendet wurde. Wenn Sie dies nicht angeben, wird der Abonnementfilter auf Kontoebene auf alle Protokollgruppen im Konto angewendet. Dieses Feld wird verwendet, um unendliche Protokollschleifen zu verhindern. Weitere Hinweise zum Problem der unendlichen Protokollschleife finden Sie unter [Verhinderung der Rekursion von Protokollen](#).

Für die Auswahlkriterien gilt eine Größenbeschränkung von 25 KB.

Für zentralisierte Protokollgruppen sind auch die folgenden Schlüsselemente enthalten. Diese Elemente können als Feldauswahlkriterien verwendet werden, um bei der Identifizierung der Quelle der Protokolldaten zu helfen und so eine detailliertere Filterung und Analyse von Metriken zu ermöglichen, die aus zentralisierten Protokollen abgeleitet wurden.

@aws .account

Dieses Feld identifiziert die AWS Konto-ID, von der das Protokollereignis stammt.

@aws .region

Dieses Feld identifiziert die AWS Region, in der das Protokollereignis generiert wurde.

Abonnementfilter auf Gruppenebene protokollieren

Sie können einen Abonnementfilter mit Amazon Kinesis Data Streams AWS Lambda, Amazon Data Firehose oder Amazon OpenSearch Service verwenden. Protokolle, die über einen Abonnementfilter an einen Service gesendet werden, sind Base64-codiert und im GZIP-Format komprimiert. Wenn Sie zentralisierte Protokolle mit Ihrem verwenden AWS Organizations, können Sie das Feld @aws . account und @aws . region System ausgeben, um zu ermitteln, welche Daten aus welchen

Konten und Regionen in Ihrer Organisation stammen. Dieser Abschnitt enthält Beispiele, anhand derer Sie einen CloudWatch Logs-Abonnementfilter erstellen können, der Protokolldaten an Firehose, Lambda, Amazon Kinesis Data Streams und Service sendet. OpenSearch

Note

Wenn Sie Ihre Protokolldaten durchsuchen möchten, finden Sie weitere Informationen unter [Filter- und Mustersyntax](#).

Beispiele

- [Beispiel 1: Abonnementfilter mit Amazon Kinesis Data Streams](#)
- [Beispiel 2: Abonnementfilter mit AWS Lambda](#)
- [Beispiel 3: Abonnementfilter mit Amazon Data Firehose](#)
- [Beispiel 4: Abonnementfilter mit Amazon OpenSearch Service](#)

Beispiel 1: Abonnementfilter mit Amazon Kinesis Data Streams

Im folgenden Beispiel wird ein Abonnementfilter mit einer Protokollgruppe verknüpft, die AWS CloudTrail -Ereignisse enthält. Der Abonnementfilter überträgt jede protokollierte Aktivität, die mit „Root“ AWS -Anmeldeinformationen ausgeführt wird, an einen Stream in Amazon Kinesis Data Streams mit dem Namen `RootAccess`. Weitere Informationen zum Senden von AWS CloudTrail Ereignissen an CloudWatch Logs finden Sie unter [Senden von CloudTrail Ereignissen an CloudWatch Logs](#) im AWS CloudTrail Benutzerhandbuch.

Note

Bevor Sie den -Stream erstellen, berechnen Sie das Volumen der generierten Protokolldaten. Vergewissern Sie sich, dass Sie einen -Stream mit einer ausreichenden Anzahl von Shards zur Bearbeitung dieses Volumens erstellen. Wenn der Stream nicht über genügend Shards verfügt, wird der Protokoll-Stream gedrosselt. Weitere Informationen zu den Volumenlimits bei Streams finden Sie unter [Kontingente und Limits](#).

Gedrosselte Zustellungen werden bis zu 24 Stunden lang neu versucht. Nach 24 Stunden werden die fehlgeschlagenen Zustellungen verworfen.

Um das Risiko der Drosselung zu verringern, können Sie die folgenden Schritte unternehmen:

- Geben Sie `randomandistribution`, wenn Sie den Abonnementfilter mit [PutSubscriptionFilter](#) oder erstellen [put-subscription-filter](#). Standardmäßig erfolgt die Verteilung des Streamfilters nach Protokolldatenströmen, was zu einer Drosselung führen kann.
- Überwachen Sie Ihren Stream mithilfe von Metriken CloudWatch . Auf diese Weise können Sie etwaige Drosselungen erkennen und Ihre Konfiguration entsprechend anpassen. Die `DeliveryThrottling` Metrik kann beispielsweise verwendet werden, um die Anzahl der Protokollereignisse nachzuverfolgen, für die CloudWatch Logs bei der Weiterleitung von Daten an das Abonnementziel gedrosselt wurde. Weitere Informationen zur Überwachung finden Sie unter [Überwachung mit CloudWatch Metriken](#).
- Verwenden Sie den On-Demand-Kapazitätsmodus für Ihren Stream in Amazon Kinesis Data Streams. Der On-Demand-Modus passt sich Ihren Workloads sofort an, wenn sie steigen oder sinken. Weitere Informationen zum On-Demand-Kapazitätsmodus finden Sie unter [On-Demand-Modus](#).
- Schränken Sie Ihr CloudWatch Abonnement-Filtermuster so ein, dass es der Kapazität Ihres Streams in Amazon Kinesis Data Streams entspricht. Wenn Sie zu viele Daten an den Stream senden, müssen Sie möglicherweise die Filtergröße reduzieren oder die Filterkriterien anpassen.

So erstellen Sie einen Abonnementfilter für Amazon Kinesis Data Streams

1. Erstellen Sie einen Ziel-Stream mit dem folgenden Befehl:

```
$ C:\> aws kinesis create-stream --stream-name "RootAccess" --shard-count 1
```

2. Warten Sie, bis der -Stream "Aktiv" wird (dies kann einige Minuten dauern). Sie können den folgenden Amazon Kinesis Data Streams Streams-Befehl [describe-stream](#) verwenden, um das zu überprüfen. `StreamDescription` `StreamStatus` `Eigentum`. Notieren Sie sich außerdem den Wert `StreamDescription.streamArn`, da Sie ihn in einem späteren Schritt benötigen werden:

```
aws kinesis describe-stream --stream-name "RootAccess"
```

Das Folgende ist eine Beispielausgabe:

```
{
```

```

"StreamDescription": {
  "StreamStatus": "ACTIVE",
  "StreamName": "RootAccess",
  "StreamARN": "arn:aws:kinesis:us-east-1:123456789012:stream/RootAccess",
  "Shards": [
    {
      "ShardId": "shardId-000000000000",
      "HashKeyRange": {
        "EndingHashKey": "340282366920938463463374607431768211455",
        "StartingHashKey": "0"
      },
      "SequenceNumberRange": {
        "StartingSequenceNumber":
          "49551135218688818456679503831981458784591352702181572610"
      }
    }
  ]
}

```

- Erstellen Sie die IAM-Rolle, die CloudWatch Logs die Berechtigung erteilt, Daten in Ihren Stream einzufügen. Zunächst müssen Sie eine Vertrauensrichtlinie in einer Datei erstellen (z. B. ~/TrustPolicyForCWL-Kinesis.json). Verwenden Sie einen Text-Editor, um diese Richtlinie zu erstellen. Verwenden Sie zum Erstellen nicht die IAM-Konsole.

Diese Richtlinie enthält einen globalen Bedingungskontextschlüssel `aws:SourceArn`, um das Confused-Deputy-Problem zu vermeiden. Weitere Informationen finden Sie unter [Confused-Deputy-Prävention](#).

```

{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringLike": { "aws:SourceArn": "arn:aws:logs:region:123456789012:*" }
    }
  }
}

```

4. Verwenden Sie den Befehl `create-role`, um die IAM-Rolle zu erstellen und die Datei mit der Vertrauensrichtlinie anzugeben. Notieren Sie den zurückgegebenen Wert `Role.Arn`, da Sie ihn in einem späteren Schritt benötigen:

```
aws iam create-role --role-name CWLtoKinesisRole --assume-role-policy-document
file://~/TrustPolicyForCWL-Kinesis.json
```

Es folgt ein Beispiel für die Ausgabe.

```
{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "logs.amazonaws.com"
        },
        "Condition": {
          "StringLike": {
            "aws:SourceArn": { "arn:aws:logs:region:123456789012:*" }
          }
        }
      }
    },
    "RoleId": "AA0IIAH450GAB4HC5F431",
    "CreateDate": "2015-05-29T13:46:29.431Z",
    "RoleName": "CWLtoKinesisRole",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisRole"
  }
}
```

5. Erstellen Sie eine Berechtigungsrichtlinie, um zu definieren, welche Aktionen CloudWatch Logs auf Ihrem Konto ausführen kann. Erstellen Sie zunächst eine Berechtigungsrichtlinie in einer Datei (z. B. `~/PermissionsForCWL-Kinesis.json`). Verwenden Sie einen Text-Editor, um diese Richtlinie zu erstellen. Verwenden Sie zum Erstellen nicht die IAM-Konsole.

```
{
  "Statement": [
    {
```

```

    "Effect": "Allow",
    "Action": "kinesis:PutRecord",
    "Resource": "arn:aws:kinesis:region:123456789012:stream/RootAccess"
  }
]
}

```

6. Verknüpfen Sie die Berechtigungsrichtlinie mit dem folgenden Befehl [put-role-policy](#) mit der Rolle:

```

aws iam put-role-policy --role-name CWLtoKinesisRole --policy-name Permissions-
Policy-For-CWL --policy-document file://~/PermissionsForCWL-Kinesis.json

```

7. Nachdem sich der Stream im Status Aktiv befindet und Sie die IAM-Rolle erstellt haben, können Sie den CloudWatch Protokoll-Abonnementfilter erstellen. Der Abonnementfilter startet sofort den Fluss der Echtzeit-Protokolldaten aus der gewählten Protokollgruppe an den -Stream:

```

aws logs put-subscription-filter \
  --log-group-name "CloudTrail/logs" \
  --filter-name "RootAccess" \
  --filter-pattern "${$.userIdentity.type = Root}" \
  --destination-arn "arn:aws:kinesis:region:123456789012:stream/RootAccess" \
  --role-arn "arn:aws:iam::123456789012:role/CWLtoKinesisRole"

```

8. Nachdem Sie den Abonnementfilter eingerichtet haben, leitet CloudWatch Logs alle eingehenden Protokollereignisse, die dem Filtermuster entsprechen, an Ihren Stream weiter. Sie können überprüfen, ob dies der Fall ist, indem Sie sich einen Amazon Kinesis Data Streams Streams-Shard-Iterator schnappen und den Amazon Kinesis Data Streams Streams-Befehl get-records verwenden, um einige Amazon Kinesis Data Streams Streams-Datensätze abzurufen:

```

aws kinesis get-shard-iterator --stream-name RootAccess --shard-id
shardId-000000000000 --shard-iterator-type TRIM_HORIZON

```

```

{
  "ShardIterator":
  "AAAAAAAAAAFGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRGB9v4scv+3vaq+f+0IK8zM5My8ID
+g6rMo7UKWeI4+IWik20Sh0uP"
}

```

```
aws kinesis get-records --limit 10 --shard-iterator "AAAAAAAAAAFGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRGB9v4scv+3vaq+f+0IK8zM5My8ID
+g6rMo7UKWeI4+IWiK20Sh0uP"
```

Beachten Sie, dass Sie diesen Aufruf möglicherweise einige Male tätigen müssen, bevor Amazon Kinesis Data Streams Daten zurückgibt.

Sie sollten davon ausgehen, dass die Antwort in einem Datensatz-Array angezeigt wird. Das Datenattribut in einem Amazon Kinesis Data Streams Streams-Datensatz ist base64-codiert und mit dem GZIP-Format komprimiert. Sie können die Rohdaten über die Befehlszeile mit den folgenden Unix-Befehlen überprüfen:

```
echo -n "<Content of Data>" | base64 -d | zcat
```

Die dekodierten und dekomprimierten base64-Daten sind als JSON mit folgender Struktur formatiert:

```
{
  "owner": "111111111111",
  "logGroup": "CloudTrail/logs",
  "logStream": "111111111111_CloudTrail/logs_us-east-1",
  "subscriptionFilters": [
    "Destination"
  ],
  "messageType": "DATA_MESSAGE",
  "logEvents": [
    {
      "id": "31953106606966983378809025079804211143289615424298221568",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\\\"Root\\\"}}",
    },
    {
      "id": "31953106606966983378809025079804211143289615424298221569",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\\\"Root\\\"}}",
    },
  ],
}
```



```
{
  "id": "31953106606966983378809025079804211143289615424298221570",
  "timestamp": 1432826855000,
  "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\n\"Root\"}}\"
}
```

Die Schlüsselemente in der oben genannten Datenstruktur sind folgende:

owner

Die AWS Konto-ID der ursprünglichen Protokolldaten.

logGroup

Der Name der Protokollgruppe der ursprünglichen Protokolldaten.

logStream

Der Name des Protokoll-Stream der ursprünglichen Protokolldaten.

subscriptionFilters

Die Namenliste der Abonnementfilter, die mit den ursprünglichen Protokolldaten übereingestimmt haben.

messageType

Datennachrichten verwenden den Typ "DATA_MESSAGE". Manchmal können CloudWatch Logs Amazon Kinesis Data Streams Streams-Datensätze mit dem Typ „CONTROL_MESSAGE“ ausgeben, hauptsächlich um zu überprüfen, ob das Ziel erreichbar ist.

logEvents

Die tatsächlichen Protokolldaten, die als Array von Protokollereignis-Datensätzen dargestellt werden. Die "id"-Eigenschaft ist eine eindeutige Kennung für jedes Protokollereignis.

Beispiel 2: Abonnementfilter mit AWS Lambda

In diesem Beispiel erstellen Sie einen CloudWatch Logs-Abonnementfilter, der Protokolldaten an Ihre AWS Lambda Funktion sendet.

 Note

Berechnen Sie das Volume der generierten Protokolldaten, bevor Sie die Lambda-Funktion erstellen. Vergewissern Sie sich, dass Sie eine Funktion erstellen, die dieses Volumen bearbeiten kann. Wenn die Funktion nicht über genügend Volumen verfügt, wird der Protokoll-Stream gedrosselt. Weitere Informationen über Lambda-Limits finden Sie unter [AWS Lambda -Limits](#).

So erstellen Sie einen Abonnementfilter für Lambda

1. Erstellen Sie die AWS Lambda Funktion.

Stellen Sie sicher, dass Sie die Lambda-Ausführungsrolle eingerichtet haben. Weitere Informationen finden Sie unter [Schritt 2.2: Erstellen einer IAM-Rolle \(Ausführungsrolle\)](#) im AWS Lambda -Entwicklerhandbuch.

2. Öffnen Sie einen Text-Editor, und erstellen Sie eine Datei mit dem Namen `helloWorld.js` mit folgendem Inhalt:

```
var zlib = require('zlib');
exports.handler = function(input, context) {
  var payload = Buffer.from(input.awslogs.data, 'base64');
  zlib.gunzip(payload, function(e, result) {
    if (e) {
      context.fail(e);
    } else {
      result = JSON.parse(result.toString());
      console.log("Event Data:", JSON.stringify(result, null, 2));
      context.succeed();
    }
  });
};
```

3. Komprimieren Sie die Datei "helloWorld.js", und speichern Sie sie unter dem Namen `helloWorld.zip`.
4. Verwenden Sie den folgenden Befehl, wobei die Rolle die Lambda-Ausführungsrolle ist, die Sie im ersten Schritt eingerichtet haben:

```
aws lambda create-function \
  --function-name helloworld \
```

```
--zip-file fileb://file-path/helloWorld.zip \
--role lambda-execution-role-arn \
--handler helloWorld.handler \
--runtime nodejs12.x
```

5. Erteilen Sie CloudWatch Logs die Erlaubnis, Ihre Funktion auszuführen. Verwenden Sie den folgenden Befehl, und ersetzen Sie den Platzhalter für das Konto mit Ihrem eigenen Konto und den Platzhalter für die Protokollgruppe mit der zu verarbeitenden Protokollgruppe:

```
aws lambda add-permission \
--function-name "helloworld" \
--statement-id "helloworld" \
--principal "logs.amazonaws.com" \
--action "lambda:InvokeFunction" \
--source-arn "arn:aws:logs:region:123456789123:log-group:TestLambda:*" \
--source-account "123456789012"
```

6. Erstellen Sie einen Abonnementfilter mit den folgenden Befehl, und ersetzen Sie den Platzhalter für das Konto mit Ihrem eigenen Konto und den Platzhalter für die Protokollgruppe mit der zu verarbeitenden Protokollgruppe:

```
aws logs put-subscription-filter \
--log-group-name myLogGroup \
--filter-name demo \
--filter-pattern "" \
--destination-arn arn:aws:lambda:region:123456789123:function:helloworld
```

7. (Optional) Führen Sie einen Test mit einem Beispiel-Protokollereignis durch. Geben Sie an der Eingabeaufforderung den folgenden Befehl ein. Damit wird eine einfache Protokollnachricht in den registrierten Stream gestellt.

Um die Ausgabe Ihrer Lambda-Funktion zu sehen, navigieren Sie zur Lambda-Funktion, wo Sie die Ausgabe in/sehen werden: `aws/lambda/helloworld`

```
aws logs put-log-events --log-group-name myLogGroup --log-stream-name stream1 --
log-events "[{\"timestamp\":<CURRENT TIMESTAMP MILLIS> , \"message\": \"Simple
Lambda Test\"}]"
```

Sie sollten davon ausgehen, dass die Antwort mit einem Lambda-Array angezeigt wird. Das Attribut `Data` im Lambda-Datensatz ist base64-kodiert und im GZIP-Format komprimiert. Die tatsächliche Nutzlast, die Lambda erhält, hat folgendes Format: `{ "awslogs": { "data":`

"BASE64ENCODED_GZIP_COMPRESSED_DATA"} }]. Sie können die Rohdaten über die Befehlszeile mit den folgenden Unix-Befehlen überprüfen:

```
echo -n "<BASE64ENCODED_GZIP_COMPRESSED_DATA>" | base64 -d | zcat
```

Die dekodierten und dekomprimierten base64-Daten sind als JSON mit folgender Struktur formatiert:

```
{
  "owner": "123456789012",
  "logGroup": "CloudTrail",
  "logStream": "123456789012_CloudTrail_us-east-1",
  "subscriptionFilters": [
    "Destination"
  ],
  "messageType": "DATA_MESSAGE",
  "logEvents": [
    {
      "id": "31953106606966983378809025079804211143289615424298221568",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\n\"Root\"}}",
    },
    {
      "id": "31953106606966983378809025079804211143289615424298221569",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\n\"Root\"}}",
    },
    {
      "id": "31953106606966983378809025079804211143289615424298221570",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\n\"Root\"}}",
    }
  ]
}
```

Die Schlüsselemente in der oben genannten Datenstruktur sind folgende:

owner

Die AWS Konto-ID der ursprünglichen Protokolldaten.

logGroup

Der Name der Protokollgruppe der ursprünglichen Protokolldaten.

logStream

Der Name des Protokoll-Stream der ursprünglichen Protokolldaten.

subscriptionFilters

Die Namenliste der Abonnementfilter, die mit den ursprünglichen Protokolldaten übereingestimmt haben.

messageType

Datennachrichten verwenden den Typ "DATA_MESSAGE". Manchmal können CloudWatch Logs Lambda-Datensätze mit dem Typ „CONTROL_MESSAGE“ ausgeben, hauptsächlich um zu überprüfen, ob das Ziel erreichbar ist.

logEvents

Die tatsächlichen Protokolldaten, die als Array von Protokollereignis-Datensätzen dargestellt werden. Die "id"-Eigenschaft ist eine eindeutige Kennung für jedes Protokollereignis.

Beispiel 3: Abonnementfilter mit Amazon Data Firehose

In diesem Beispiel erstellen Sie ein CloudWatch Logs-Abonnement, das alle eingehenden Protokollereignisse, die Ihren definierten Filtern entsprechen, an Ihren Amazon Data Firehose-Lieferstream sendet. Daten, die von CloudWatch Logs an Amazon Data Firehose gesendet werden, sind bereits mit GZIP-Komprimierung der Stufe 6 komprimiert, sodass Sie in Ihrem Firehose-Lieferstream keine Komprimierung verwenden müssen. Anschließend können Sie die Dekomprimierungsfunktion in Firehose verwenden, um die Protokolle automatisch zu dekomprimieren. Weitere Informationen finden Sie unter [CloudWatch Protokolle an Firehose senden](#).

Note

Bevor Sie den Firehose erstellen, berechnen Sie das Volumen der Protokolldaten, die generiert werden. Stellen Sie sicher, dass Sie einen Firehose-Stream erstellen, der dieses

Volume verarbeiten kann. Wenn der Stream das Volumen nicht bearbeiten kann, wird die Protokoll-Stream gedrosselt. Weitere Informationen zu den Volumenbeschränkungen für Firehose-Streams finden Sie unter [Amazon Data Firehose Data Limits](#).

Um einen Abonnementfilter für Firehose zu erstellen

1. Erstellen Sie einen Amazon-Simple-Storage-Service-(Amazon S3)-Bucket. Wir empfehlen Ihnen, einen Bucket zu verwenden, der speziell für CloudWatch Logs erstellt wurde. Wenn Sie jedoch einen vorhandenen Bucket verwenden möchten, gehen Sie direkt zu Schritt 2.

Führen Sie den folgenden Befehl aus, und ersetzen Sie die Platzhalter-Region mit der Region, die Sie verwenden möchten:

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket2 --create-bucket-configuration  
LocationConstraint=region
```

Das Folgende ist eine Beispielausgabe:

```
{  
  "Location": "/amzn-s3-demo-bucket2"  
}
```

2. Erstellen Sie die IAM-Rolle, die Amazon Data Firehose die Erlaubnis erteilt, Daten in Ihren Amazon S3 S3-Bucket zu legen.

Weitere Informationen finden Sie unter [Zugriffskontrolle mit Amazon Data Firehose](#) im Amazon Data Firehose Developer Guide.

Verwenden Sie zunächst einen Text-Editor zum Erstellen einer Vertrauensrichtlinie in einer Datei `~/TrustPolicyForFirehose.json` wie folgt:

```
{  
  "Statement": {  
    "Effect": "Allow",  
    "Principal": { "Service": "firehose.amazonaws.com" },  
    "Action": "sts:AssumeRole"  
  }  
}
```

3. Verwenden Sie den Befehl `create-role`, um die IAM-Rolle zu erstellen und die Datei mit der Vertrauensrichtlinie anzugeben. Notieren Sie den zurückgegebenen Wert `Role.Arn`, da Sie ihn in einem späteren Schritt benötigen:

```
aws iam create-role \
  --role-name FirehoseToS3Role \
  --assume-role-policy-document file://~/TrustPolicyForFirehose.json

{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "firehose.amazonaws.com"
        }
      }
    },
    "RoleId": "AA0IIAH450GAB4HC5F431",
    "CreateDate": "2015-05-29T13:46:29.431Z",
    "RoleName": "FirehoseToS3Role",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:role/FirehoseToS3Role"
  }
}
```

4. Erstellen Sie eine Berechtigungsrichtlinie, um zu definieren, welche Aktionen Firehose auf Ihrem Konto ausführen kann. Verwenden Sie zunächst einen Text-Editor zum Erstellen einer Berechtigungsrichtlinie in einer Datei `~/PermissionsForFirehose.json`:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:PutObject" ],
    }
  ]
}
```

```

    "Resource": [
      "arn:aws:s3:::amzn-s3-demo-bucket2",
      "arn:aws:s3:::amzn-s3-demo-bucket2/*" ]
    }
  ]
}

```

5. Verknüpfen Sie die Berechtigungsrichtlinie mit dem folgenden Befehl `put-role-policy` mit der Rolle:

```

aws iam put-role-policy --role-name FirehoseToS3Role --policy-name Permissions-
Policy-For-Firehose --policy-document file://~/PermissionsForFirehose.json

```

6. Erstellen Sie wie folgt einen Firehose-Ziel-Lieferstream und ersetzen Sie dabei die Platzhalterwerte für RoleARN und BucketARN durch die Rolle und den Bucket, die Sie erstellt haben: ARNs

```

aws firehose create-delivery-stream \
  --delivery-stream-name 'my-delivery-stream' \
  --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::123456789012:role/FirehoseToS3Role", "BucketARN":
  "arn:aws:s3:::amzn-s3-demo-bucket2"}'

```

Beachten Sie, dass Firehose automatisch ein Präfix im YYYY/MM/DD/HH UTC-Zeitformat für gelieferte Amazon S3 S3-Objekte verwendet. Sie können ein zusätzliches Präfix vor dem Zeitformat-Präfix hinzufügen. Wenn das Präfix mit einem Schrägstrich (/) endet, wird es als Ordner im Amazon-S3-Bucket angezeigt.

7. Warten Sie, bis der Stream aktiv wird (dies kann einige Minuten dauern). Sie können den `describe-delivery-stream` Befehl Firehose verwenden, um das `DeliveryStreamDescription` zu überprüfen. `DeliveryStreamStatus` Eigentum. Beachten Sie außerdem die `DeliveryStreamDescription`. `DeliveryStreamARN`-Wert, wie Sie ihn in einem späteren Schritt benötigen werden:

```

aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
{
  "DeliveryStreamDescription": {
    "HasMoreDestinations": false,
    "VersionId": "1",
    "CreateTimestamp": 1446075815.822,

```



```

    "DeliveryStreamARN": "arn:aws:firehose:us-
east-1:123456789012:deliverystream/my-delivery-stream",
    "DeliveryStreamStatus": "ACTIVE",
    "DeliveryStreamName": "my-delivery-stream",
    "Destinations": [
      {
        "DestinationId": "destinationId-000000000001",
        "S3DestinationDescription": {
          "CompressionFormat": "UNCOMPRESSED",
          "EncryptionConfiguration": {
            "NoEncryptionConfig": "NoEncryption"
          },
          "RoleARN": "delivery-stream-role",
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket2",
          "BufferingHints": {
            "IntervalInSeconds": 300,
            "SizeInMBs": 5
          }
        }
      }
    ]
  }
}

```

8. Erstellen Sie die IAM-Rolle, die CloudWatch Logs die Berechtigung erteilt, Daten in Ihren Firehose-Lieferstream aufzunehmen. Verwenden Sie zunächst einen Text-Editor zum Erstellen einer Vertrauensrichtlinie in einer Datei `~/TrustPolicyForCWL.json`:

Diese Richtlinie enthält einen globalen Bedingungskontextschlüssel `aws:SourceArn`, um das Confused-Deputy-Problem zu vermeiden. Weitere Informationen finden Sie unter [Confused-Deputy-Prävention](#).

```

{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringLike": {
        "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
      }
    }
  }
}

```

```
}
```

9. Verwenden Sie den Befehl `create-role`, um die IAM-Rolle zu erstellen und die Datei mit der Vertrauensrichtlinie anzugeben. Notieren Sie den zurückgegebenen Wert `Role.Arn`, da Sie ihn in einem späteren Schritt benötigen:

```
aws iam create-role \
--role-name CWLtoKinesisFirehoseRole \
--assume-role-policy-document file://~/TrustPolicyForCWL.json

{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "logs.amazonaws.com"
        },
        "Condition": {
          "StringLike": {
            "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
          }
        }
      }
    },
    "RoleId": "AAOIIAH450GAB4HC5F431",
    "CreateDate": "2015-05-29T13:46:29.431Z",
    "RoleName": "CWLtoKinesisFirehoseRole",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
  }
}
```

10. Erstellen Sie eine Berechtigungsrichtlinie, um zu definieren, welche Aktionen CloudWatch Logs auf Ihrem Konto ausführen kann. Verwenden Sie zunächst einen Text-Editor zum Erstellen einer Berechtigungsrichtliniendatei (beispielsweise `~/PermissionsForCWL.json`):

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["firehose:PutRecord"],
```

```

    "Resource": [
      "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-
name"]
    ]
  }
}

```

11. Verknüpfen Sie die Berechtigungsrichtlinie mit dem Befehl `put-role-policy` mit der Rolle:

```

aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-
name Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json

```

12. Nachdem sich der Amazon Data Firehose-Lieferstream im aktiven Zustand befindet und Sie die IAM-Rolle erstellt haben, können Sie den CloudWatch Logs-Abonnementfilter erstellen. Der Abonnementfilter startet sofort den Fluss von Echtzeit-Protokolldaten von der ausgewählten Protokollgruppe zu Ihrem Amazon Data Firehose-Lieferstream:

```

aws logs put-subscription-filter \
  --log-group-name "CloudTrail" \
  --filter-name "Destination" \
  --filter-pattern "{$.userIdentity.type = Root}" \
  --destination-arn "arn:aws:firehose:region:123456789012:deliverystream/my-
delivery-stream" \
  --role-arn "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"

```

13. Nachdem Sie den Abonnementfilter eingerichtet haben, leitet CloudWatch Logs alle eingehenden Protokollereignisse, die dem Filtermuster entsprechen, an Ihren Amazon Data Firehose-Lieferstream weiter. Ihre Daten werden in Ihrem Amazon S3 basierend auf dem Zeitpufferintervall angezeigt, das in Ihrem Amazon Data Firehose-Lieferstream festgelegt wurde. Sobald genügend Zeit abgelaufen ist, können Sie die Daten im Amazon-S3-Bucket überprüfen.

```

aws s3api list-objects --bucket 'amzn-s3-demo-bucket2' --prefix 'firehose/'
{
  "Contents": [
    {
      "LastModified": "2015-10-29T00:01:25.000Z",
      "ETag": "\"a14589f8897f4089d3264d9e2d1f1610\"",
      "StorageClass": "STANDARD",
      "Key": "firehose/2015/10/29/00/my-delivery-stream-2015-10-29-00-01-21-
a188030a-62d2-49e6-b7c2-b11f1a7ba250",
      "Owner": {
        "DisplayName": "cloudwatch-logs",

```

```

        "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b5"
      },
      "Size": 593
    },
    {
      "LastModified": "2015-10-29T00:35:41.000Z",
      "ETag": "\"a7035b65872bb2161388ffb63dd1aec5\"",
      "StorageClass": "STANDARD",
      "Key": "firehose/2015/10/29/00/my-delivery-
stream-2015-10-29-00-35-40-7cc92023-7e66-49bc-9fd4-fc9819cc8ed3",
      "Owner": {
        "DisplayName": "cloudwatch-logs",
        "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b6"
      },
      "Size": 5752
    }
  ]
}

```

```

aws s3api get-object --bucket 'amzn-s3-demo-bucket2' --key 'firehose/2015/10/29/00/
my-delivery-stream-2015-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250'
testfile.gz

```

```

{
  "AcceptRanges": "bytes",
  "ContentType": "application/octet-stream",
  "LastModified": "Thu, 29 Oct 2015 00:07:06 GMT",
  "ContentLength": 593,
  "Metadata": {}
}

```

Die Daten im Amazon-S3-Objekt sind im GZIP-Format komprimiert. Sie können die Rohdaten über die Befehlszeile mit dem folgenden Unix-Befehl überprüfen:

```
zcat testfile.gz
```

Beispiel 4: Abonnementfilter mit Amazon OpenSearch Service

In diesem Beispiel erstellen Sie ein CloudWatch Logs-Abonnement, das eingehende Protokollereignisse, die Ihren definierten Filtern entsprechen, an Ihre OpenSearch Service-Domain sendet.

Um einen Abonnementfilter für OpenSearch Service zu erstellen

1. Erstellen Sie eine OpenSearch Dienstdomäne. Weitere Informationen finden Sie unter [OpenSearch Dienstdomänen erstellen](#)
2. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
3. Wählen Sie im Navigationsbereich Protokollgruppen aus.
4. Wählen Sie den Namen der Protokollgruppe aus.
5. Wählen Sie Aktionen, Abonnementfilter, Amazon OpenSearch Service-Abonnementfilter erstellen.
6. Wählen Sie aus, ob Sie zu einem Cluster in diesem Konto oder einem anderen Konto streamen möchten.
 - Wenn Sie Dieses Konto ausgewählt haben, wählen Sie die Domain aus, die Sie in Schritt 1 erstellt haben.
 - Wenn Sie Anderes Konto ausgewählt haben, geben Sie ARN und Endpunkt dieser Domain ein.
7. Wenn Sie ein anderes Konto gewählt haben, geben Sie den Domänen-ARN und den Endpunkt an.
8. Wählen Sie für Amazon OpenSearch Service Cluster den Namen des Clusters aus, in den die Protokollgruppendaten geliefert werden.
9. Wählen Sie ein Protokollformat.
10. Geben Sie unter Filtermuster für Abonnements die Begriffe oder das Muster ein, nach denen Sie in Ihren Protokollereignissen suchen möchten. Dadurch wird sichergestellt, dass Sie nur die Daten, die Sie interessieren, an Ihren OpenSearch Service-Cluster senden. Weitere Informationen finden Sie unter [Filtermustersyntax für metrische Filter](#).
11. (Optional) Wählen Sie unter Select log data to test (Zu testende Protokolldaten auswählen) einen Protokoll-Stream und anschließend Test pattern (Muster testen) aus, um zu überprüfen, ob Ihr Suchfilter die erwarteten Ergebnisse zurückgeben.
12. Wählen Sie dann Start Streaming (Streamen starten).

Abonnementfilter auf Kontoebene

Important

Es besteht die Gefahr, dass bei Abonnementfiltern eine endlose rekursive Schleife entsteht, die zu einem starken Anstieg der Datenerfassung führen kann, wenn sie nicht behoben wird. Um dieses Risiko zu minimieren, empfehlen wir Ihnen, in Ihren Abonnementfiltern auf Kontoebene Auswahlkriterien zu verwenden, um Protokollgruppen auszuschließen, die Protokolldaten aus Ressourcen aufnehmen, die Teil des Workflows für die Abonnementzustellung sind. Weitere Informationen zu diesem Problem und zur Bestimmung der auszuschließenden Protokollgruppen finden Sie unter [Verhinderung der Rekursion von Protokollen](#)

Sie können eine Abonnementrichtlinie auf Kontoebene einrichten, die eine Teilmenge der Protokollgruppen im Konto umfasst. Die Kontoabonnementrichtlinie kann mit Amazon Kinesis Data Streams oder Amazon Data Firehose funktionieren. AWS Lambda Protokolle, die über eine Abonnementrichtlinie auf Kontoebene an einen Service gesendet werden, sind Base64-kodiert und im GZIP-Format komprimiert. Dieser Abschnitt enthält Beispiele, denen Sie folgen können, um ein Abonnement auf Kontoebene für Amazon Kinesis Data Streams, Lambda und Firehose zu erstellen.

Note

Um eine Liste aller Abonnementfilterrichtlinien in Ihrem Konto anzuzeigen, verwenden Sie den `describe-account-policies` Befehl mit dem Wert von `SUBSCRIPTION_FILTER_POLICY` für den Parameter. Weitere Informationen finden Sie unter [describe-account-policies](#).

Beispiele

- [Beispiel 1: Abonnementfilter mit Amazon Kinesis Data Streams](#)
- [Beispiel 2: Abonnementfilter mit AWS Lambda](#)
- [Beispiel 3: Abonnementfilter mit Amazon Data Firehose](#)

Beispiel 1: Abonnementfilter mit Amazon Kinesis Data Streams

Bevor Sie einen Amazon Kinesis Data Streams Streams-Datenstream erstellen, der mit einer Abonnementrichtlinie auf Kontoebene verwendet werden soll, berechnen Sie das Volumen der Protokolldaten, die generiert werden. Vergewissern Sie sich, dass Sie einen -Stream mit einer ausreichenden Anzahl von Shards zur Bearbeitung dieses Volumens erstellen. Wenn ein Stream nicht über genügend Shards verfügt, wird er gedrosselt. Weitere Informationen zu Volumenbeschränkungen für Streams finden Sie unter [Kontingente und Limits](#) in der Amazon Kinesis Data Streams Streams-Dokumentation.

Warning

Da die Protokollereignisse mehrerer Protokollgruppen an das Ziel weitergeleitet werden, besteht die Gefahr einer Drosselung. Gedrosselte Zustellungen werden bis zu 24 Stunden lang neu versucht. Nach 24 Stunden werden die fehlgeschlagenen Zustellungen verworfen. Um das Risiko der Drosselung zu verringern, können Sie die folgenden Schritte unternehmen:

- Überwachen Sie Ihren Amazon Kinesis Data Streams Streams-Stream mit CloudWatch Metriken. Auf diese Weise können Sie Drosselungen erkennen und Ihre Konfiguration entsprechend anpassen. Die `DeliveryThrottling` Metrik verfolgt beispielsweise die Anzahl der Protokollereignisse, für die CloudWatch Logs bei der Weiterleitung von Daten an das Abonnementziel gedrosselt wurde. Weitere Informationen finden Sie unter [Überwachung mit CloudWatch Metriken](#).
- Verwenden Sie den On-Demand-Kapazitätsmodus für Ihren Stream in Amazon Kinesis Data Streams. Der On-Demand-Modus passt sich Ihren Workloads sofort an, wenn sie steigen oder sinken. Weitere Informationen finden Sie unter [On-Demand-Modus](#).
- Schränken Sie Ihr CloudWatch Logs-Abonnement-Filtermuster so ein, dass es der Kapazität Ihres Streams in Amazon Kinesis Data Streams entspricht. Wenn Sie zu viele Daten an den Stream senden, müssen Sie möglicherweise die Filtergröße reduzieren oder die Filterkriterien anpassen.

Das folgende Beispiel verwendet eine Abonnementrichtlinie auf Kontoebene, um alle Protokollereignisse an einen Stream in Amazon Kinesis Data Streams weiterzuleiten. Das Filtermuster ordnet alle Protokollereignisse dem Text zu `Test` und leitet sie an den Stream in Amazon Kinesis Data Streams weiter.

So erstellen Sie eine Abonnementrichtlinie auf Kontoebene für Amazon Kinesis Data Streams

1. Erstellen Sie einen Ziel--Stream mit dem folgenden Befehl:

```
$ C:\> aws kinesis create-stream --stream-name "TestStream" --shard-count 1
```

2. Warten Sie einige Minuten, bis der Stream aktiv wird. Sie können überprüfen, ob der Stream aktiv ist, indem Sie den Befehl [describe-stream](#) verwenden, um das zu überprüfen. StreamDescription StreamStatusEigentum.

```
aws kinesis describe-stream --stream-name "TestStream"
```

Das Folgende ist eine Beispielausgabe:

```
{
  "StreamDescription": {
    "StreamStatus": "ACTIVE",
    "StreamName": "TestStream",
    "StreamARN": "arn:aws:kinesis:region:123456789012:stream/TestStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "EXAMPLE8463463374607431768211455",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
            "EXAMPLE688818456679503831981458784591352702181572610"
        }
      }
    ]
  }
}
```

3. Erstellen Sie die IAM-Rolle, die CloudWatch Logs die Erlaubnis erteilt, Daten in Ihren Stream einzufügen. Zunächst müssen Sie eine Vertrauensrichtlinie in einer Datei erstellen (z. B. ~/TrustPolicyForCWL-Kinesis.json). Verwenden Sie einen Text-Editor, um diese Richtlinie zu erstellen.

Diese Richtlinie enthält einen globalen Bedingungskontextschlüssel `aws:SourceArn`, um das Confused-Deputy-Problem zu vermeiden. Weitere Informationen finden Sie unter [Confused-Deputy-Prävention](#).

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringLike": { "aws:SourceArn": "arn:aws:logs:region:123456789012:*" }
    }
  }
}
```

4. Verwenden Sie den Befehl `create-role`, um die IAM-Rolle zu erstellen und die Datei mit der Vertrauensrichtlinie anzugeben. Notieren Sie den zurückgegebenen Wert `RoleArn`, da Sie ihn in einem späteren Schritt benötigen:

```
aws iam create-role --role-name CWLtoKinesisRole --assume-role-policy-document
file:///~/TrustPolicyForCWL-Kinesis.json
```

Es folgt ein Beispiel für die Ausgabe.

```
{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "logs.amazonaws.com"
        },
        "Condition": {
          "StringLike": {
            "aws:SourceArn": { "arn:aws:logs:region:123456789012:*" }
          }
        }
      }
    },
    "RoleId": "EXAMPLE450GAB4HC5F431",
```

```

    "CreateDate": "2023-05-29T13:46:29.431Z",
    "RoleName": "CWLtoKinesisRole",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisRole"
  }
}

```

5. Erstellen Sie eine Berechtigungsrichtlinie, um zu definieren, welche Aktionen CloudWatch Logs auf Ihrem Konto ausführen kann. Erstellen Sie zunächst eine Berechtigungsrichtlinie in einer Datei (z. B. `~/PermissionsForCWL-Kinesis.json`). Verwenden Sie einen Text-Editor, um diese Richtlinie zu erstellen. Verwenden Sie nicht die IAM-Konsole, um es zu erstellen.

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:123456789012:stream/TestStream"
    }
  ]
}

```

6. Verknüpfen Sie die Berechtigungsrichtlinie mit dem folgenden Befehl [put-role-policy](#) mit der Rolle:

```

aws iam put-role-policy --role-name CWLtoKinesisRole --policy-name Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL-Kinesis.json

```

7. Nachdem sich der Stream im Status Aktiv befindet und Sie die IAM-Rolle erstellt haben, können Sie die Abonnementfilterrichtlinie für CloudWatch Logs erstellen. Die Richtlinie startet sofort den Fluss von Protokolldaten in Echtzeit zu Ihrem Stream. In diesem Beispiel ERROR werden alle Protokollereignisse, die die Zeichenfolge enthalten, gestreamt, mit Ausnahme der Ereignisse in den Protokollgruppen mit dem Namen `LogGroupToExclude1` und `LogGroupToExclude2`.

```

aws logs put-account-policy \
  --policy-name "ExamplePolicy" \
  --policy-type "SUBSCRIPTION_FILTER_POLICY" \
  --policy-document '{"RoleArn":"arn:aws:iam::123456789012:role/CWLtoKinesisRole", "DestinationArn":"arn:aws:kinesis:region:123456789012:stream/TestStream", "FilterPattern": "Test", "Distribution": "Random"}' \
  --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1", "LogGroupToExclude2"]' \

```

```
--scope "ALL"
```

8. Nachdem Sie den Abonnementfilter eingerichtet haben, leitet CloudWatch Logs alle eingehenden Protokollereignisse, die dem Filtermuster und den Auswahlkriterien entsprechen, an Ihren Stream weiter.

Das `selection-criteria` Feld ist optional, aber wichtig, um Protokollgruppen, die zu einer unendlichen Protokollrekursion führen können, aus einem Abonnementfilter auszuschließen. Weitere Informationen zu diesem Problem und zur Bestimmung, welche Protokollgruppen ausgeschlossen werden sollen, finden Sie unter [Verhinderung der Rekursion von Protokollen](#). Derzeit ist NOT IN der einzige unterstützte Operator für `selection-criteria`.

Sie können den Fluss der Protokollereignisse überprüfen, indem Sie einen Amazon Kinesis Data Streams Streams-Shard-Iterator verwenden und den Amazon Kinesis Data Streams `get-records` Streams-Befehl verwenden, um einige Amazon Kinesis Data Streams Streams-Datensätze abzurufen:

```
aws kinesis get-shard-iterator --stream-name TestStream --shard-id
shardId-000000000000 --shard-iterator-type TRIM_HORIZON
```

```
{
  "ShardIterator":
  "AAAAAAAAAAFGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+0IK8zM5My8ID
+g6rMo7UKWeI4+IWiK20Sh0uP"
}
```

```
aws kinesis get-records --limit 10 --shard-iterator "AAAAAAAAAAFGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+0IK8zM5My8ID
+g6rMo7UKWeI4+IWiK20Sh0uP"
```

Möglicherweise müssen Sie diesen Befehl einige Male verwenden, bevor Amazon Kinesis Data Streams Daten zurückgibt.

Sie sollten davon ausgehen, dass die Antwort in einem Datensatz-Array angezeigt wird. Das Datenattribut in einem Amazon Kinesis Data Streams Streams-Datensatz ist base64-codiert und mit dem GZIP-Format komprimiert. Sie können die Rohdaten über die Befehlszeile mit den folgenden Unix-Befehlen überprüfen:

```
echo -n "<Content of Data>" | base64 -d | zcat
```

Die dekodierten und dekomprimierten base64-Daten sind als JSON mit folgender Struktur formatiert:

```
{
  "messageType": "DATA_MESSAGE",
  "owner": "123456789012",
  "logGroup": "Example1",
  "logStream": "logStream1",
  "subscriptionFilters": [
    "ExamplePolicy"
  ],
  "logEvents": [
    {
      "id": "31953106606966983378809025079804211143289615424298221568",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\n\"Root\"}}",
    },
    {
      "id": "31953106606966983378809025079804211143289615424298221569",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\n\"Root\"}}",
    },
    {
      "id": "31953106606966983378809025079804211143289615424298221570",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\n\"Root\"}}",
    }
  ],
  "policyLevel": "ACCOUNT_LEVEL_POLICY"
}
```

Die wichtigsten Elemente der Datenstruktur sind die folgenden:

`messageType`

Datennachrichten verwenden den Typ "DATA_MESSAGE". Manchmal geben CloudWatch Logs Amazon Kinesis Data Streams Streams-Datensätze mit dem Typ „CONTROL_MESSAGE“ aus, hauptsächlich um zu überprüfen, ob das Ziel erreichbar ist.

`owner`

Die AWS Konto-ID der ursprünglichen Protokolldaten.

`logGroup`

Der Name der Protokollgruppe der ursprünglichen Protokolldaten.

`logStream`

Der Name des Protokoll-Stream der ursprünglichen Protokolldaten.

`subscriptionFilters`

Die Namenliste der Abonnementfilter, die mit den ursprünglichen Protokolldaten übereingestimmt haben.

`logEvents`

Die tatsächlichen Protokolldaten, die als Array von Protokollereignis-Datensätzen dargestellt werden. Die "id"-Eigenschaft ist eine eindeutige Kennung für jedes Protokollereignis.

`Richtlinienebene`

Die Ebene, auf der die Richtlinie durchgesetzt wurde. „ACCOUNT_LEVEL_POLICY“ ist die Filterrichtlinie `policyLevel` für Abonnements auf Kontoebene.

Beispiel 2: Abonnementfilter mit AWS Lambda

In diesem Beispiel erstellen Sie eine Abonnementfilterrichtlinie für CloudWatch Logs auf Kontoebene, die Protokolldaten an Ihre AWS Lambda Funktion sendet.

Warning

Berechnen Sie das Volume der generierten Protokolldaten, bevor Sie die Lambda-Funktion erstellen. Vergewissern Sie sich, dass Sie eine Funktion erstellen, die dieses Volumen

bearbeiten kann. Wenn die Funktion das Volumen nicht verarbeiten kann, wird der Log-Stream gedrosselt. Da die Protokollereignisse aller Protokollgruppen oder einer Teilmenge der Protokollgruppen des Kontos an das Ziel weitergeleitet werden, besteht die Gefahr einer Drosselung. Weitere Informationen über Lambda-Limits finden Sie unter [AWS Lambda - Limits](#).

So erstellen Sie eine Abonnementfilterrichtlinie auf Kontoebene für Lambda

1. Erstellen Sie die Funktion. AWS Lambda

Stellen Sie sicher, dass Sie die Lambda-Ausführungsrolle eingerichtet haben. Weitere Informationen finden Sie unter [Schritt 2.2: Erstellen einer IAM-Rolle \(Ausführungsrolle\)](#) im AWS Lambda -Entwicklerhandbuch.

2. Öffnen Sie einen Text-Editor, und erstellen Sie eine Datei mit dem Namen `helloWorld.js` mit folgendem Inhalt:

```
var zlib = require('zlib');
exports.handler = function(input, context) {
  var payload = Buffer.from(input.awslogs.data, 'base64');
  zlib.gunzip(payload, function(e, result) {
    if (e) {
      context.fail(e);
    } else {
      result = JSON.parse(result.toString());
      console.log("Event Data:", JSON.stringify(result, null, 2));
      context.succeed();
    }
  });
};
```

3. Komprimieren Sie die Datei "helloWorld.js", und speichern Sie sie unter dem Namen `helloWorld.zip`.
4. Verwenden Sie den folgenden Befehl, wobei die Rolle die Lambda-Ausführungsrolle ist, die Sie im ersten Schritt eingerichtet haben:

```
aws lambda create-function \
  --function-name helloworld \
  --zip-file fileb://file-path/helloWorld.zip \
  --role lambda-execution-role-arn \
```

```
--handler helloWorld.handler \  
--runtime nodejs18.x
```

5. Erteilen Sie CloudWatch Logs die Erlaubnis, Ihre Funktion auszuführen. Verwenden Sie den folgenden Befehl und ersetzen Sie das Platzhalterkonto durch Ihr eigenes Konto.

```
aws lambda add-permission \  
  --function-name "helloworld" \  
  --statement-id "helloworld" \  
  --principal "logs.amazonaws.com" \  
  --action "lambda:InvokeFunction" \  
  --source-arn "arn:aws:logs:region:123456789012:log-group:*" \  
  --source-account "123456789012"
```

6. Erstellen Sie mit dem folgenden Befehl eine Abonnementfilterrichtlinie auf Kontoebene und ersetzen Sie dabei das Platzhalterkonto durch Ihr eigenes Konto. In diesem Beispiel ERROR werden alle Protokollereignisse, die die Zeichenfolge enthalten, gestreamt, mit Ausnahme der Ereignisse in den Protokollgruppen mit dem Namen und. LogGroupToExclude1 LogGroupToExclude2

```
aws logs put-account-policy \  
  --policy-name "ExamplePolicyLambda" \  
  --policy-type "SUBSCRIPTION_FILTER_POLICY" \  
  --policy-document  
'{"DestinationArn":"arn:aws:lambda:region:123456789012:function:helloWorld",  
  "FilterPattern": "Test", "Distribution": "Random"}' \  
  --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",  
  "LogGroupToExclude2"]' \  
  --scope "ALL"
```

Nachdem Sie den Abonnementfilter eingerichtet haben, leitet CloudWatch Logs alle eingehenden Protokollereignisse, die dem Filtermuster und den Auswahlkriterien entsprechen, an Ihren Stream weiter.

Das `selection-criteria` Feld ist optional, aber wichtig, um Protokollgruppen, die zu einer unendlichen Protokollrekursion führen können, aus einem Abonnementfilter auszuschließen. Weitere Informationen zu diesem Problem und zur Bestimmung, welche Protokollgruppen ausgeschlossen werden sollen, finden Sie unter [Verhinderung der Rekursion von Protokollen](#). Derzeit ist NOT IN der einzige unterstützte Operator für `selection-criteria`.

7. (Optional) Führen Sie einen Test mit einem Beispiel-Protokollereignis durch. Geben Sie an der Eingabeaufforderung den folgenden Befehl ein. Damit wird eine einfache Protokollnachricht in den registrierten Stream gestellt.

Um die Ausgabe Ihrer Lambda-Funktion zu sehen, navigieren Sie zur Lambda-Funktion, wo Sie die Ausgabe in/sehen werden: `aws/lambda/helloworld`

```
aws logs put-log-events --log-group-name Example1 --log-stream-name logStream1 --
log-events "[{"timestamp\":"CURRENT_TIMESTAMP_MILLIS", \"message\": \"Simple Lambda
Test\"}]"
```

Sie sollten davon ausgehen, dass die Antwort mit einem Lambda-Array angezeigt wird. Das Attribut `Data` im Lambda-Datensatz ist base64-kodiert und im GZIP-Format komprimiert. Die tatsächliche Nutzlast, die Lambda erhält, hat folgendes Format: `{ "awslogs": { "data": "BASE64ENCODED_GZIP_COMPRESSED_DATA" } }`. Sie können die Rohdaten über die Befehlszeile mit den folgenden Unix-Befehlen überprüfen:

```
echo -n "<BASE64ENCODED_GZIP_COMPRESSED_DATA>" | base64 -d | zcat
```

Die dekodierten und dekomprimierten base64-Daten sind als JSON mit folgender Struktur formatiert:

```
{
  "messageType": "DATA_MESSAGE",
  "owner": "123456789012",
  "logGroup": "Example1",
  "logStream": "logStream1",
  "subscriptionFilters": [
    "ExamplePolicyLambda"
  ],
  "logEvents": [
    {
      "id": "31953106606966983378809025079804211143289615424298221568",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\", \"userIdentity\":{\"type\": \"Root\"}}",
    },
    {
      "id": "31953106606966983378809025079804211143289615424298221569",
      "timestamp": 1432826855000,
```



```

      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\\\"Root\\\"}"}
    },
    {
      "id": "31953106606966983378809025079804211143289615424298221570",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\\\"Root\\\"}"}
    }
  ],
  "policyLevel": "ACCOUNT_LEVEL_POLICY"
}

```

Note

Der Abonnementfilter auf Kontoebene wird nicht auf die Protokollgruppe der Ziel-Lambda-Funktion angewendet. Dadurch soll eine unendliche Protokollrekursion verhindert werden, die zu einer Erhöhung der Abrechnung der Datenerfassung führen kann. Weitere Informationen zu diesem Problem finden Sie unter. [Verhinderung der Rekursion von Protokollen](#)

Die wichtigsten Elemente der Datenstruktur sind die folgenden:

messageType

Datennachrichten verwenden den Typ "DATA_MESSAGE". Manchmal geben CloudWatch Logs Amazon Kinesis Data Streams Streams-Datensätze mit dem Typ „CONTROL_MESSAGE“ aus, hauptsächlich um zu überprüfen, ob das Ziel erreichbar ist.

owner

Die AWS Konto-ID der ursprünglichen Protokolldaten.

logGroup

Der Name der Protokollgruppe der ursprünglichen Protokolldaten.

logStream

Der Name des Protokoll-Stream der ursprünglichen Protokolldaten.

subscriptionFilters

Die Namenliste der Abonnementfilter, die mit den ursprünglichen Protokolldaten übereingestimmt haben.

logEvents

Die tatsächlichen Protokolldaten, die als Array von Protokollereignis-Datensätzen dargestellt werden. Die "id"-Eigenschaft ist eine eindeutige Kennung für jedes Protokollereignis.

Richtlinienebene

Die Ebene, auf der die Richtlinie durchgesetzt wurde. „ACCOUNT_LEVEL_POLICY“ ist die Filterrichtlinie `policyLevel` für Abonnements auf Kontoebene.

Beispiel 3: Abonnementfilter mit Amazon Data Firehose

In diesem Beispiel erstellen Sie eine CloudWatch Logs-Abonnementfilterrichtlinie auf Kontoebene, die eingehende Protokollereignisse, die Ihren definierten Filtern entsprechen, an Ihren Amazon Data Firehose-Lieferstream sendet. Daten, die von CloudWatch Logs an Amazon Data Firehose gesendet werden, sind bereits mit GZIP-Komprimierung der Stufe 6 komprimiert, sodass Sie in Ihrem Firehose-Lieferstream keine Komprimierung verwenden müssen. Anschließend können Sie die Dekomprimierungsfunktion in Firehose verwenden, um die Protokolle automatisch zu dekomprimieren. Weitere Informationen finden Sie unter [Mithilfe CloudWatch von Protokollen in Kinesis Data Firehose schreiben](#).

Warning

Bevor Sie den Firehose erstellen, berechnen Sie das Volumen der Protokolldaten, die generiert werden. Stellen Sie sicher, dass Sie einen Firehose-Stream erstellen, der dieses Volume verarbeiten kann. Wenn der Stream das Volumen nicht bearbeiten kann, wird die Protokoll-Stream gedrosselt. Weitere Informationen zu den Volumenbeschränkungen für Firehose-Streams finden Sie unter [Amazon Data Firehose Data Limits](#).

Um einen Abonnementfilter für Firehose zu erstellen

1. Erstellen Sie einen Amazon-Simple-Storage-Service-(Amazon S3)-Bucket. Wir empfehlen Ihnen, einen Bucket zu verwenden, der speziell für CloudWatch Logs erstellt wurde. Wenn Sie jedoch einen vorhandenen Bucket verwenden möchten, gehen Sie direkt zu Schritt 2.

Führen Sie den folgenden Befehl aus, und ersetzen Sie die Platzhalter-Region mit der Region, die Sie verwenden möchten:

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket2 --create-bucket-configuration
LocationConstraint=region
```

Das Folgende ist eine Beispielausgabe:

```
{
  "Location": "/amzn-s3-demo-bucket2"
}
```

2. Erstellen Sie die IAM-Rolle, die Amazon Data Firehose die Erlaubnis erteilt, Daten in Ihren Amazon S3 S3-Bucket zu legen.

Weitere Informationen finden Sie unter [Zugriffskontrolle mit Amazon Data Firehose](#) im Amazon Data Firehose Developer Guide.

Verwenden Sie zunächst einen Text-Editor zum Erstellen einer Vertrauensrichtlinie in einer Datei `~/TrustPolicyForFirehose.json` wie folgt:

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "firehose.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

3. Verwenden Sie den Befehl `create-role`, um die IAM-Rolle zu erstellen und die Datei mit der Vertrauensrichtlinie anzugeben. Notieren Sie sich den zurückgegebenen `Role.Arn`-Wert, da Sie ihn in einem späteren Schritt benötigen werden:

```
aws iam create-role \
  --role-name FirehosetoS3Role \
  --assume-role-policy-document file://~/TrustPolicyForFirehose.json

{
  "Role": {
    "AssumeRolePolicyDocument": {
```

```

        "Statement": {
            "Action": "sts:AssumeRole",
            "Effect": "Allow",
            "Principal": {
                "Service": "firehose.amazonaws.com"
            }
        },
        "RoleId": "EXAMPLE50GAB4HC5F431",
        "CreateDate": "2023-05-29T13:46:29.431Z",
        "RoleName": "FirehoseToS3Role",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/FirehoseToS3Role"
    }
}

```

4. Erstellen Sie eine Berechtigungsrichtlinie, um zu definieren, welche Aktionen Firehose auf Ihrem Konto ausführen kann. Verwenden Sie zunächst einen Text-Editor zum Erstellen einer Berechtigungsrichtlinie in einer Datei `~/PermissionsForFirehose.json`:

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:PutObject" ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket2",
        "arn:aws:s3:::amzn-s3-demo-bucket2/*" ]
    }
  ]
}

```

5. Verknüpfen Sie die Berechtigungsrichtlinie mit dem folgenden Befehl `put-role-policy` mit der Rolle:

```
aws iam put-role-policy --role-name FirehoseToS3Role --policy-name Permissions-Policy-For-Firehose --policy-document file:///~/PermissionsForFirehose.json
```

6. Erstellen Sie wie folgt einen Firehose-Ziel-Lieferstream und ersetzen Sie dabei die Platzhalterwerte für RoleARN und BucketARN durch die Rolle und den Bucket, die Sie erstellt haben: ARNs

```
aws firehose create-delivery-stream \
  --delivery-stream-name 'my-delivery-stream' \
  --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::123456789012:role/FirehoseToS3Role", "BucketARN":  
  "arn:aws:s3:::amzn-s3-demo-bucket2"}'
```

Firehose verwendet automatisch ein Präfix im YYYY/MM/DD/HH UTC-Zeitformat für gelieferte Amazon S3 S3-Objekte. Sie können ein zusätzliches Präfix vor dem Zeitformat-Präfix hinzufügen. Wenn das Präfix mit einem Schrägstrich (/) endet, wird es als Ordner im Amazon-S3-Bucket angezeigt.

7. Warten Sie ein paar Minuten, bis der Stream aktiv wird. Sie können den describe-delivery-stream-Befehl Firehose verwenden, um das DeliveryStreamDescription zu überprüfen. DeliveryStreamStatusEigenschaft. Beachten Sie außerdem die DeliveryStreamDescription.DeliveryStreamARN-Wert, wie Sie ihn in einem späteren Schritt benötigen werden:

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
{
  "DeliveryStreamDescription": {
    "HasMoreDestinations": false,
    "VersionId": "1",
    "CreateTimestamp": 1446075815.822,
    "DeliveryStreamARN": "arn:aws:firehose:us-east-1:123456789012:deliverystream/my-delivery-stream",
    "DeliveryStreamStatus": "ACTIVE",
    "DeliveryStreamName": "my-delivery-stream",
    "Destinations": [
      {
        "DestinationId": "destinationId-000000000001",
        "S3DestinationDescription": {
          "CompressionFormat": "UNCOMPRESSED",
          "EncryptionConfiguration": {
            "NoEncryptionConfig": "NoEncryption"
          }
        }
      }
    ]
  }
}
```

```

    },
    "RoleARN": "delivery-stream-role",
    "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket2",
    "BufferingHints": {
        "IntervalInSeconds": 300,
        "SizeInMBs": 5
    }
  }
]
}
}

```

8. Erstellen Sie die IAM-Rolle, die CloudWatch Logs die Berechtigung erteilt, Daten in Ihren Firehose-Lieferstream aufzunehmen. Verwenden Sie zunächst einen Text-Editor zum Erstellen einer Vertrauensrichtlinie in einer Datei `~/TrustPolicyForCWL.json`:

Diese Richtlinie enthält einen globalen Bedingungskontextschlüssel `aws:SourceArn`, um das Confused-Deputy-Problem zu vermeiden. Weitere Informationen finden Sie unter [Confused-Deputy-Prävention](#).

```

{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringLike": {
        "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
      }
    }
  }
}

```

9. Verwenden Sie den Befehl `create-role`, um die IAM-Rolle zu erstellen und die Datei mit der Vertrauensrichtlinie anzugeben. Notieren Sie sich den zurückgegebenen `Role.Arn`-Wert, da Sie ihn in einem späteren Schritt benötigen werden:

```

aws iam create-role \
--role-name CWLtoKinesisFirehoseRole \
--assume-role-policy-document file:///~/TrustPolicyForCWL.json

```

```
{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "logs.amazonaws.com"
        },
        "Condition": {
          "StringLike": {
            "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
          }
        }
      }
    },
    "RoleId": "AA0IIAH450GAB4HC5F431",
    "CreateDate": "2015-05-29T13:46:29.431Z",
    "RoleName": "CWLtoKinesisFirehoseRole",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
  }
}
```

10. Erstellen Sie eine Berechtigungsrichtlinie, um zu definieren, welche Aktionen CloudWatch Logs auf Ihrem Konto ausführen können. Verwenden Sie zunächst einen Text-Editor zum Erstellen einer Berechtigungsrichtliniendatei (beispielsweise ~/PermissionsForCWL.json):

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["firehose:PutRecord"],
      "Resource": [
        "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-  
name"]
      }
    ]
  }
}
```

11. Verknüpfen Sie die Berechtigungsrichtlinie mit dem Befehl `put-role-policy` mit der Rolle:

```
aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-
name Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json
```

12. Nachdem sich der Amazon Data Firehose-Lieferstream im aktiven Status befindet und Sie die IAM-Rolle erstellt haben, können Sie die Abonnementfilterrichtlinie CloudWatch Logs auf Kontoebene erstellen. Die Richtlinie startet sofort den Fluss von Protokolldaten in Echtzeit von der ausgewählten Protokollgruppe zu Ihrem Amazon Data Firehose-Lieferstream:

```
aws logs put-account-policy \
  --policy-name "ExamplePolicyFirehose" \
  --policy-type "SUBSCRIPTION_FILTER_POLICY" \
  --policy-document '{"RoleArn":"arn:aws:iam::123456789012:role/
CWLtoKinesisFirehoseRole", "DestinationArn":"arn:aws:firehose:us-
east-1:123456789012:deliverystream/delivery-stream-name", "FilterPattern": "Test",
"Distribution": "Random"}' \
  --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
"LogGroupToExclude2"]' \
  --scope "ALL"
```

13. Nachdem Sie den Abonnementfilter eingerichtet haben, leitet CloudWatch Logs die eingehenden Protokollereignisse, die dem Filtermuster entsprechen, an Ihren Amazon Data Firehose-Lieferstream weiter.

Das `selection-criteria` Feld ist optional, aber wichtig, um Protokollgruppen, die zu einer unendlichen Protokollrekursion führen können, aus einem Abonnementfilter auszuschließen. Weitere Informationen zu diesem Problem und zur Bestimmung, welche Protokollgruppen ausgeschlossen werden sollen, finden Sie unter [Verhinderung der Rekursion von Protokollen](#). Derzeit ist NOT IN der einzige unterstützte Operator für `selection-criteria`.

Ihre Daten werden in Ihrem Amazon S3 basierend auf dem Zeitpufferintervall angezeigt, das in Ihrem Amazon Data Firehose-Lieferstream festgelegt wurde. Sobald genügend Zeit abgelaufen ist, können Sie die Daten im Amazon-S3-Bucket überprüfen.

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket2' --prefix 'firehose/'
{
  "Contents": [
    {
      "LastModified": "2023-10-29T00:01:25.000Z",
      "ETag": "\"a14589f8897f4089d3264d9e2d1f1610\"",
      "StorageClass": "STANDARD",
```



```

      "Key": "firehose/2015/10/29/00/my-delivery-stream-2015-10-29-00-01-21-
a188030a-62d2-49e6-b7c2-b11f1a7ba250",
      "Owner": {
        "DisplayName": "cloudwatch-logs",
        "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b5"
      },
      "Size": 593
    },
    {
      "LastModified": "2015-10-29T00:35:41.000Z",
      "ETag": "\"a7035b65872bb2161388ffb63dd1aec5\"",
      "StorageClass": "STANDARD",
      "Key": "firehose/2023/10/29/00/my-delivery-stream-2023-10-29-00-35-40-
EXAMPLE-7e66-49bc-9fd4-fc9819cc8ed3",
      "Owner": {
        "DisplayName": "cloudwatch-logs",
        "ID": "EXAMPLE6be062b19584e0b7d84ecc19237f87b6"
      },
      "Size": 5752
    }
  ]
}

```

```

aws s3api get-object --bucket 'amzn-s3-demo-bucket2' --key 'firehose/2023/10/29/00/
my-delivery-stream-2023-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250'
testfile.gz

```

```

{
  "AcceptRanges": "bytes",
  "ContentType": "application/octet-stream",
  "LastModified": "Thu, 29 Oct 2023 00:07:06 GMT",
  "ContentLength": 593,
  "Metadata": {}
}

```

Die Daten im Amazon-S3-Objekt sind im GZIP-Format komprimiert. Sie können die Rohdaten über die Befehlszeile mit dem folgenden Unix-Befehl überprüfen:

```
zcat testfile.gz
```

Kontoübergreifende, regionsübergreifende Abonnements

Sie können mit einem Inhaber eines anderen AWS Kontos zusammenarbeiten und dessen Protokollereignisse auf Ihren AWS Ressourcen empfangen, z. B. in einem Amazon Kinesis- oder Amazon Data Firehose (dies wird als kontoübergreifender Datenaustausch bezeichnet). Diese Protokollereignisdaten können beispielsweise aus einem zentralisierten Amazon Kinesis Data Streams oder Firehose-Stream gelesen werden, um eine benutzerdefinierte Verarbeitung und Analyse durchzuführen. Benutzerdefinierte Verarbeitung ist besonders nützlich, wenn Sie über viele Konten zusammenarbeiten und Daten analysieren.

Wenn beispielsweise die IT-Sicherheitsgruppe eines Unternehmens Daten zur Echtzeit-Erkennung von Eindringversuchen oder auf Unregelmäßigkeiten analysieren möchten, kann sie eine Prüfung aller Konten in allen Abteilungen des Unternehmens durchführen und dabei alle Produktionsprotokolle für eine zentrale Verarbeitung erfassen. Ein Echtzeit-Stream von Ereignisdaten für diese Konten kann zusammengestellt und an die Informationssicherheitsgruppen übermittelt werden, die Amazon Kinesis Data Streams verwenden können, um die Daten an ihre bestehenden Sicherheitsanalysesysteme anzuhängen.

Note

Die Protokollgruppe und das Ziel müssen sich in derselben AWS Region befinden. Die AWS -Ressource, auf die das Ziel verweist, kann sich jedoch in einer anderen Region befinden. In den Beispielen in den folgenden Abschnitten werden alle regionsspezifischen Ressourcen in USA Ost (Nord-Virginia) erstellt.

Wenn Sie Mitgliedskonten konfiguriert haben AWS Organizations und mit ihnen arbeiten, können Sie die Protokollzentralisierung verwenden, um Protokolldaten von Quellkonten in einem zentralen Überwachungskonto zu sammeln.

Wenn Sie mit zentralisierten Protokollgruppen arbeiten, können Sie beim Erstellen von Abonnementfiltern die folgenden Systemfelddimensionen verwenden:

- `@aws.account`- Diese Dimension stellt die AWS Konto-ID dar, von der das Protokollereignis ausgegangen ist.
- `@aws.region`- Diese Dimension stellt die AWS Region dar, in der das Protokollereignis generiert wurde.

Diese Dimensionen helfen bei der Identifizierung der Quelle der Protokolldaten und ermöglichen so eine detailliertere Filterung und Analyse von Metriken, die aus zentralisierten Protokollen stammen.

Themen

- [Kontoübergreifender regionsübergreifender Austausch von Protokolldaten mithilfe von Amazon Kinesis Data Streams](#)
- [Kontoübergreifender regionsübergreifender Austausch von Protokolldaten mit Firehose](#)
- [Kontoübergreifende, regionsübergreifende Abonnements auf Kontoebene mit Amazon Kinesis Data Streams](#)
- [Kontoübergreifende, regionsübergreifende Abonnements auf Kontoebene mit Firehose](#)

Kontoübergreifender regionsübergreifender Austausch von Protokolldaten mithilfe von Amazon Kinesis Data Streams

Wenn Sie ein kontoübergreifendes Abonnement erstellen, können Sie ein einzelnes Konto oder eine Organisation angeben, die der Sender sein soll. Wenn Sie eine Organisation angeben, können alle Konten in der Organisation mit diesem Verfahren Protokolle an das Receiver-Konto senden.

Wenn Sie Protokolldaten für mehrere Konten freigeben möchten, müssen Sie Sender und Receiver der Protokolldaten festlegen:

- Absender der Protokolldaten — Ruft die Zielinformationen vom Empfänger ab und teilt Logs mit, dass CloudWatch Logs bereit ist, seine Protokollereignisse an das angegebene Ziel zu senden. In den Verfahren im Rest dieses Abschnitts wird der Absender der Protokolldaten mit der fiktiven AWS Kontonummer 1111111111 angezeigt.

Wenn mehrere Konten in einer Organisation Protokolle an ein Empfängerkonto senden, können Sie eine Richtlinie erstellen, die allen Konten in der Organisation die Berechtigung zum Senden von Protokollen an das Empfängerkonto gewährt. Sie müssen immer noch separate Abonnementfilter für jedes Senderkonto einrichten.

- Empfänger von Protokolldaten — richtet ein Ziel ein, das einen Amazon Kinesis Data Streams Streams-Stream kapselt und CloudWatch Logs darüber informiert, dass der Empfänger Protokolldaten empfangen möchte. Der Empfänger gibt dann die Informationen über sein Ziel für den Absender frei. In den Verfahren im Rest dieses Abschnitts wird der Empfänger der Protokolldaten mit der fiktiven AWS Kontonummer 999999999999 angezeigt.

Um Protokollereignisse von kontoübergreifenden Benutzern zu empfangen, erstellt der Empfänger der Protokolldaten zunächst ein Protokollziel. CloudWatch Jedes Ziel umfasst die folgenden Schlüsselemente:

Name des Ziels

Der Name der Zielregion, die Sie erstellen möchten.

Ziel-ARN

Der Amazon-Ressourcenname (ARN) der AWS Ressource, die Sie als Ziel für den Abonnement-Feed verwenden möchten.

ARN der Rolle

Eine AWS Identity and Access Management (IAM) -Rolle, die CloudWatch Logs die erforderlichen Berechtigungen erteilt, um Daten in den ausgewählten Stream zu übertragen.

Zugriffsrichtlinie

Ein IAM-Richtliniendokument (im JSON-Format, das in der IAM-Richtliniengrammatik geschrieben ist), das kontrolliert, welche Benutzer Schreibberechtigung für das Ziel haben.

Note

Die Protokollgruppe und das Ziel müssen sich in derselben AWS Region befinden. Die AWS Ressource, auf die das Ziel verweist, kann sich jedoch in einer anderen Region befinden. In den Beispielen in den folgenden Abschnitten werden alle regionsspezifischen Ressourcen in USA Ost (Nord-Virginia) erstellt.

Themen

- [Einrichten eines neuen kontoübergreifenden Abonnements](#)
- [Aktualisieren eines bestehenden kontoübergreifenden Abonnements](#)

Einrichten eines neuen kontoübergreifenden Abonnements

Befolgen Sie die Schritte in diesen Abschnitten, um ein neues kontoübergreifendes Protokollabonnement einzurichten.

Themen

- [Schritt 1: Erstellen eines Ziels](#)
- [Schritt 2: \(Nur bei Verwendung einer Organisation\) Erstellen Sie eine IAM-Rolle](#)
- [Schritt 3: Add/validate IAM-Berechtigungen für das kontoübergreifende Ziel](#)
- [Schritt 4: Erstellen eines Abonnementfilters](#)
- [Validierung des Ablaufs von Protokollereignissen](#)
- [Ändern der Mitgliedschaft im Ziel zur Laufzeit](#)

Schritt 1: Erstellen eines Ziels

Important

Alle Schritte in diesem Verfahren müssen im Konto des Protokolldatenempfängers ausgeführt werden.

In diesem Beispiel hat das Konto des Empfängers der Protokolldaten die Konto-ID 9999999999, während die AWS Konto-ID des Absenders der Protokolldaten 1111111111 lautet. AWS

In diesem Beispiel wird ein Ziel mithilfe eines Amazon Kinesis Data Streams-Streams namens und einer Rolle erstellt RecipientStream, die es CloudWatch Logs ermöglicht, Daten darauf zu schreiben.

Wenn das Ziel erstellt ist, sendet CloudWatch Logs im Namen des Empfängerkontos eine Testnachricht an das Ziel. Wenn der Abonnementfilter später aktiv ist, sendet CloudWatch Logs im Namen des Quellkontos Protokollereignisse an das Ziel.

So erstellen Sie ein Ziel

1. Erstellen Sie im Empfängerkonto einen Ziel-Stream in Amazon Kinesis Data Streams. Geben Sie an der Eingabeaufforderung Folgendes ein:

```
aws kinesis create-stream --stream-name "RecipientStream" --shard-count 1
```

2. Warten Sie, bis der -Stream aktiv wird. Sie können den Befehl `aws kinesis describe-stream` verwenden, um das zu überprüfen. `StreamDescription` `StreamStatus` `Eigentum`. Notieren Sie sich außerdem den Wert `StreamDescription.streamArn`, da Sie ihn später an Logs übergeben werden: CloudWatch

```
aws kinesis describe-stream --stream-name "RecipientStream"
{
  "StreamDescription": {
    "StreamStatus": "ACTIVE",
    "StreamName": "RecipientStream",
    "StreamARN": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "34028236692093846346337460743176EXAMPLE",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
"4955113521868881845667950383198145878459135270218EXAMPLE"
        }
      }
    ]
  }
}
```

Es kann einige Minuten dauern, bis der Stream im aktiven Status angezeigt wird.

- Erstellen Sie die IAM-Rolle, die CloudWatch Logs die Berechtigung erteilt, Daten in Ihren Stream einzufügen. Zunächst müssen Sie eine Vertrauensrichtlinie in einer Datei `~/TrustPolicyForCWL.json` erstellen. Erstellen Sie in einem Text-Editor die Richtliniendatei, verwenden Sie nicht die IAM-Konsole.

Diese Richtlinie enthält einen globalen Bedingungskontextschlüssel `aws:SourceArn`, der das `sourceAccountId` angibt, um das Confused-Deputy-Problem zu vermeiden. Wenn Sie die Quell-Kontonummer beim ersten Aufruf noch nicht kennen, empfehlen wir Ihnen, die Ziel-ARN in das Quell-ARN-Feld einzutragen. Bei den folgenden Aufrufen sollten Sie als Quell-ARN den tatsächlichen Quell-ARN angeben, den Sie beim ersten Aufruf ermittelt haben. Weitere Informationen finden Sie unter [Confused-Deputy-Prävention](#).

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "Service": "logs.amazonaws.com"
    }
  }
}
```

```

    },
    "Condition": {
      "StringLike": {
        "aws:SourceArn": [
          "arn:aws:logs:region:sourceAccountId:*",
          "arn:aws:logs:region:recipientAccountId:"
        ]
      }
    },
    "Action": "sts:AssumeRole"
  }
}

```

4. Verwenden Sie den Befehl `aws iam create-role`, um die IAM-Rolle zu erstellen und die Vertrauensrichtlinie anzugeben. Notieren Sie sich den zurückgegebenen `Role.Arn`-Wert, da er später auch an Logs übergeben wird: CloudWatch

```

aws iam create-role \
--role-name CWLtoKinesisRole \
--assume-role-policy-document file://~/TrustPolicyForCWL.json

{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Condition": {
          "StringLike": {
            "aws:SourceArn": [
              "arn:aws:logs:region:sourceAccountId:*",
              "arn:aws:logs:region:recipientAccountId:"
            ]
          }
        },
        "Principal": {
          "Service": "logs.amazonaws.com"
        }
      }
    },
    "RoleId": "AA0IIAH450GAB4HC5F431",
    "CreateDate": "2015-05-29T13:46:29.431Z",
    "RoleName": "CWLtoKinesisRole",

```

```

    "Path": "/",
    "Arn": "arn:aws:iam::999999999999:role/CWLtoKinesisRole"
  }
}

```

5. Erstellen Sie eine Berechtigungsrichtlinie, um zu definieren, welche Aktionen CloudWatch Logs auf Ihrem Konto ausführen kann. Verwenden Sie zunächst einen Texteditor, um eine Berechtigungsrichtlinie in einer Datei `~/PermissionsForCWL.json` zu erstellen:

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:999999999999:stream/RecipientStream"
    }
  ]
}

```

6. Ordnen Sie die Berechtigungsrichtlinie der Rolle zu, indem Sie den Befehl `aws iam` verwenden: `put-role-policy`

```

aws iam put-role-policy \
  --role-name CWLtoKinesisRole \
  --policy-name Permissions-Policy-For-CWL \
  --policy-document file:///~/PermissionsForCWL.json

```

7. Nachdem sich der Stream im aktiven Status befindet und Sie die IAM-Rolle erstellt haben, können Sie das CloudWatch Logs-Ziel erstellen.
 - a. In diesem Schritt wird keine Zugriffsrichtlinie mit Ihrem Ziel verknüpft. Es ist zudem erst der erste von zwei Schritten zum Erstellen eines Ziels. Notieren Sie sich den in der Nutzlast ausgegebenen Wert `DestinationArn`:

```

aws logs put-destination \
  --destination-name "testDestination" \
  --target-arn "arn:aws:kinesis:region:999999999999:stream/RecipientStream" \
  --role-arn "arn:aws:iam::999999999999:role/CWLtoKinesisRole"

{
  "DestinationName" : "testDestination",
  "RoleArn" : "arn:aws:iam::999999999999:role/CWLtoKinesisRole",

```



```

    "DestinationArn" : "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
    "TargetArn" : "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream"
  }

```

- b. Verknüpfen Sie, nachdem Schritt 7a abgeschlossen ist, im Empfängerkonto der Protokolldaten eine Zugriffsrichtlinie mit dem Ziel. Diese Richtlinie muss die PutSubscriptionFilter Aktion logs: spezifizieren und erteilt dem Absenderkonto die Erlaubnis, auf das Ziel zuzugreifen.

Die Richtlinie erteilt dem AWS Konto, das Protokolle sendet, die entsprechenden Berechtigungen. Sie können nur dieses eine Konto in der Richtlinie angeben, oder wenn das Senderkonto Mitglied einer Organisation ist, kann die Richtlinie die Organisations-ID der Organisation angeben. Auf diese Weise können Sie nur eine Richtlinie erstellen, mit der mehrere Konten in einer Organisation Protokolle an dieses Zielkonto senden können.

Verwenden Sie einen Texteditor, um eine Datei mit dem Namen `~/AccessPolicy.json` mit einer der folgenden Richtlinienanweisungen zu erstellen.

Diese erste Beispielrichtlinie ermöglicht es allen Konten in der Organisation, die die ID o-1234567890 haben, Protokolle an das Empfängerkonto zu senden.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "logs:PutSubscriptionFilter",
      "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": [
            "o-1234567890"
          ]
        }
      }
    }
  ]
}

```

```
}
]
}
```

In diesem nächsten Beispiel kann nur das Protokolldatenabsenderkonto (111111111111) Protokolle an das Protokolldatenempfängerkonto senden.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": "logs:PutSubscriptionFilter",
      "Resource": "arn:aws:logs:us-east-1:999999999999:destination:testDestination"
    }
  ]
}
```

- c. Fügen Sie die Richtlinie, die Sie im vorherigen Schritt erstellt haben, dem Ziel an.

```
aws logs put-destination-policy \
  --destination-name "testDestination" \
  --access-policy file://~/AccessPolicy.json
```

Diese Zugriffsrichtlinie ermöglicht es Benutzern im AWS Konto mit der ID 111111111111, PutSubscriptionFilter gegen das Ziel mit ARN `arn:aws:logs::999999999999:destination:TestDestination` anzurufen. *region* Jeder Versuch PutSubscriptionFilter eines anderen Benutzers, für dieses Ziel anzurufen, wird abgewiesen.

Informationen dazu, wie Sie die Berechtigungen eines Benutzers mit einer Zugriffsrichtlinie prüfen, finden Sie unter [Verwenden der Richtlinienvvalidierung](#) im IAM-Benutzerhandbuch.

Wenn Sie fertig sind und AWS Organizations für Ihre kontoübergreifenden Berechtigungen verwenden, folgen Sie den Schritten unter [Schritt 2: \(Nur bei Verwendung einer Organisation\) Erstellen Sie eine IAM-Rolle](#). Wenn Sie dem anderen Konto Berechtigungen direkt erteilen, anstatt Organizations zu verwenden, können Sie diesen Schritt überspringen und mit fortfahren [Schritt 4: Erstellen eines Abonnementfilters](#).

Schritt 2: (Nur bei Verwendung einer Organisation) Erstellen Sie eine IAM-Rolle

Wenn Sie im vorherigen Abschnitt das Ziel mithilfe einer Zugriffsrichtlinie erstellt haben, die der Organisation, in der sich Konto 11111111111 befindet, Berechtigungen erteilt, anstatt Konto 11111111111 direkt Berechtigungen zu erteilen, führen Sie die Schritte in diesem Abschnitt aus. Andernfalls können Sie mit [Schritt 4: Erstellen eines Abonnementfilters](#) fortfahren.

Mit den Schritten in diesem Abschnitt wird eine IAM-Rolle erstellt, die davon ausgehen und überprüfen CloudWatch kann, ob das Absenderkonto berechtigt ist, einen Abonnementfilter für das Empfängerziel zu erstellen.

Führen Sie die Schritte in diesem Abschnitt im Sender-Konto aus. Die Rolle muss im Sender-Konto vorhanden sein, und Sie geben den ARN dieser Rolle im Abonnementfilter an. In diesem Beispiel ist das Sender-Konto 11111111111.

Um die IAM-Rolle zu erstellen, die für kontoübergreifende Protokollabonnements erforderlich ist, verwenden Sie AWS Organizations

1. Erstellen Sie die folgende Vertrauensrichtlinie in einer Datei namens /TrustPolicyForCWLSubscriptionFilter.json. Erstellen Sie in einem Text-Editor die Richtliniendatei, verwenden Sie nicht die IAM-Konsole.

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

2. Erstellen Sie als Nächstes die IAM-Rolle, die diese Richtlinie verwendet. Notieren Sie sich den Arn-Wert, der vom Befehl zurückgegeben wird, Sie benötigen ihn später in diesem Verfahren. In diesem Beispiel verwenden wir CWLtoSubscriptionFilterRole für den Namen der Rolle, die wir erstellen.

```
aws iam create-role \
  --role-name CWLtoSubscriptionFilterRole \
  --assume-role-policy-document file://~/
TrustPolicyForCWLSubscriptionFilter.json
```

3. Erstellen Sie eine Berechtigungsrichtlinie, um die Aktionen zu definieren, die CloudWatch Logs auf Ihrem Konto ausführen kann.
 - a. Verwenden Sie zunächst einen Text-Editor, um die folgende Berechtigungsrichtlinie in einer Datei mit dem Namen zu erstellen `~/PermissionsForCWLSubscriptionFilter.json`.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:PutLogEvents",
      "Resource": "arn:aws:logs:region:111111111111:log-
group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
    }
  ]
}
```

- b. Geben Sie den folgenden Befehl ein, um die soeben erstellte Berechtigungsrichtlinie mit der Rolle zu verknüpfen, die Sie in Schritt 2 erstellt haben.

```
aws iam put-role-policy
  --role-name CWLtoSubscriptionFilterRole
  --policy-name Permissions-Policy-For-CWL-Subscription-filter
  --policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

Wenn Sie fertig sind, können Sie mit [Schritt 4: Erstellen eines Abonnementfilters](#) fortfahren.

Schritt 3: Add/validate IAM-Berechtigungen für das kontoübergreifende Ziel

Gemäß der Bewertungslogik für AWS kontenübergreifende Richtlinien muss für den Zugriff auf eine kontoübergreifende Ressource (z. B. ein Kinesis- oder Firehose-Stream, der als Ziel für einen Abonnementfilter verwendet wird) eine identitätsbasierte Richtlinie im sendenden Konto vorhanden sein, die expliziten Zugriff auf die kontenübergreifende Zielressource gewährt. Weitere Informationen zur Logik der Richtlinienbewertung finden Sie unter [Bewertungslogik für kontenübergreifende Richtlinien](#).

Sie können die identitätsbasierte Richtlinie der IAM-Rolle oder dem IAM-Benutzer zuordnen, die Sie zum Erstellen des Abonnementfilters verwenden. Diese Richtlinie muss im übermittelnden Konto vorhanden sein. Wenn Sie die Administratorrolle verwenden, um den Abonnementfilter zu erstellen, können Sie diesen Schritt überspringen und mit [Schritt 4: Erstellen eines Abonnementfilters](#) fortfahren.

Um die IAM-Berechtigungen hinzuzufügen oder zu validieren, die für kontoübergreifende Konten erforderlich sind

1. Geben Sie den folgenden Befehl ein, um zu überprüfen, welche IAM-Rolle oder welcher IAM-Benutzer zur Ausführung von Log-Befehlen verwendet wird. AWS

```
aws sts get-caller-identity
```

Daraufhin erhalten Sie ein Ergebnis, das dem hier dargestellten entspricht:

```
{
  "UserId": "User ID",
  "Account": "sending account id",
  "Arn": "arn:aws:sending account id:role/user:RoleName/UserName"
}
```

Notieren Sie sich den Wert, der durch *RoleName* oder dargestellt wird. *UserName*

2. Melden Sie sich AWS-Managementkonsole im sendenden Konto an und suchen Sie nach den angehängten Richtlinien mit der IAM-Rolle oder dem IAM-Benutzer, die in der Ausgabe des in Schritt 1 eingegebenen Befehls zurückgegeben wurden.
3. Stellen Sie sicher, dass die mit dieser Rolle oder diesem Benutzer verknüpften Richtlinien explizite Zugriffsberechtigungen für die `logs:PutSubscriptionFilter` auf der kontoübergreifenden Zielressource enthalten.

Die folgende Richtlinie gewährt Berechtigungen zum Erstellen eines Abonnementfilters für jede Zielressource nur in einem einzigen AWS Konto, einem Konto: 999999999999

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```

    {
      "Sid": "AllowSubscriptionFiltersOnAccountResources",
      "Effect": "Allow",
      "Action": "logs:PutSubscriptionFilter",
      "Resource": [
        "arn:aws:logs:*:*:log-group:*",
        "arn:aws:logs:*:123456789012:destination:*"
      ]
    }
  ]
}

```

Die folgende Richtlinie gewährt Berechtigungen zum Erstellen eines Abonnementfilters nur für eine bestimmte Zielressource mit `sampleDestination` dem Namen „AWS Einzelkonto“`123456789012`:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSubscriptionFiltersonAccountResource",
      "Effect": "Allow",
      "Action": "logs:PutSubscriptionFilter",
      "Resource": [
        "arn:aws:logs:*:*:log-group:*",
        "arn:aws:logs:*:123456789012:destination:sampleDestination"
      ]
    }
  ]
}

```

Schritt 4: Erstellen eines Abonnementfilters

Wenn Sie ein Ziel erstellt haben, kann das Empfängerkonto der Protokolldaten den Ziel-ARN (`arn:aws:logs:us-east-1:999999999999:destination:testDestination`) für andere AWS -Konten freigeben, damit diese Protokollereignisse an dasselbe Ziel senden können. Die anderen sendenden Kontobenutzer erstellen einen Abonnementfilter für die jeweiligen Protokollgruppen für dieses

Ziel. Der Abonnementfilter startet sofort den Fluss der Echtzeit-Protokolldaten aus der gewählten Protokollgruppe an das angegebene Ziel.

Note

Wenn Sie einer gesamten Organisation Berechtigungen für den Abonnementfilter gewähren, müssen Sie den ARN der IAM-Rolle verwenden, die Sie in [Schritt 2: \(Nur bei Verwendung einer Organisation\) Erstellen Sie eine IAM-Rolle](#) erstellt haben.

Im folgenden Beispiel wird ein Abonnementfilter in einem sendenden Konto erstellt. Der Filter ist einer Protokollgruppe zugeordnet, die AWS CloudTrail Ereignisse enthält, sodass jede protokollierte Aktivität, die mit „Root“ AWS -Anmeldeinformationen ausgeführt wird, an das Ziel gesendet wird, das Sie zuvor erstellt haben. Dieses Ziel kapselt einen Stream namens `RecipientStream`.

Bei den restlichen Schritten in den folgenden Abschnitten wird davon ausgegangen, dass Sie die Anweisungen unter [CloudTrail Ereignisse an CloudWatch Protokolle senden](#) im AWS CloudTrail Benutzerhandbuch befolgt und eine Protokollgruppe erstellt haben, die Ihre CloudTrail Ereignisse enthält. Diese Schritte gehen davon aus, dass der Name dieser Protokollgruppe `CloudTrail/logs` ist.

Wenn Sie den folgenden Befehl eingeben, stellen Sie sicher, dass Sie als IAM-Benutzer angemeldet sind oder die IAM-Rolle verwenden, für die Sie die Richtlinie hinzugefügt haben, in [Schritt 3: Add/validate IAM-Berechtigungen für das kontoübergreifende Ziel](#).

```
aws logs put-subscription-filter \
  --log-group-name "CloudTrail/logs" \
  --filter-name "RecipientStream" \
  --filter-pattern "${.userIdentity.type = Root}" \
  --destination-arn "arn:aws:logs:region:999999999999:destination:testDestination"
```

Die Protokollgruppe und das Ziel müssen sich in derselben AWS Region befinden. Das Ziel kann jedoch auf eine AWS Ressource verweisen, z. B. auf einen Amazon Kinesis Data Streams Stream, der sich in einer anderen Region befindet.

Validierung des Ablaufs von Protokollereignissen

Nachdem Sie den Abonnementfilter erstellt haben, leitet CloudWatch Logs alle eingehenden Protokollereignisse, die dem Filtermuster entsprechen, an den Stream weiter, der im Zielstream mit

dem Namen "" gekapselt ist. RecipientStream Der Besitzer des Ziels kann überprüfen, ob dies der Fall ist, indem er den get-shard-iterator Befehl aws kinesis verwendet, um einen Amazon Kinesis Data Streams Streams-Shard abzurufen, und den Befehl aws kinesis get-records verwendet, um einige Amazon Kinesis Data Streams Streams-Datensätze abzurufen:

```
aws kinesis get-shard-iterator \
    --stream-name RecipientStream \
    --shard-id shardId-000000000000 \
    --shard-iterator-type TRIM_HORIZON

{
  "ShardIterator":
  "AAAAAAAAAAGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRgB9v4scv+3vaq+f
+0IK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
}

aws kinesis get-records \
    --limit 10 \
    --shard-iterator
    "AAAAAAAAAAGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRgB9v4scv+3vaq+f
+0IK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
```

Note

Möglicherweise müssen Sie den Befehl get-records einige Male erneut ausführen, bevor Amazon Kinesis Data Streams Daten zurückgibt.

Sie sollten eine Antwort mit einer Reihe von Amazon Kinesis Data Streams Streams-Datensätzen sehen. Das Datenattribut im Amazon Kinesis Data Streams Streams-Datensatz wird im GZIP-Format komprimiert und anschließend base64-codiert. Sie können die Rohdaten über die Befehlszeile mit dem folgenden Unix-Befehl überprüfen:

```
echo -n "<Content of Data>" | base64 -d | zcat
```

Die dekodierten und dekomprimierten base64-Daten sind als JSON mit folgender Struktur formatiert:


```
{
  "owner": "111111111111",
  "logGroup": "CloudTrail/logs",
  "logStream": "111111111111_CloudTrail/logs_us-east-1",
  "subscriptionFilters": [
    "RecipientStream"
  ],
  "messageType": "DATA_MESSAGE",
  "logEvents": [
    {
      "id": "3195310660696698337880902507980421114328961542429EXAMPLE",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
    }\",
    {
      "id": "3195310660696698337880902507980421114328961542429EXAMPLE",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
    }\",
    {
      "id": "3195310660696698337880902507980421114328961542429EXAMPLE",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
    }
  ]
}
```

Die wichtigsten Elemente in dieser Datenstruktur lauten wie folgt:

owner

Die AWS Konto-ID der ursprünglichen Protokolldaten.

logGroup

Der Name der Protokollgruppe der ursprünglichen Protokolldaten.

logStream

Der Name des Protokoll-Stream der ursprünglichen Protokolldaten.

subscriptionFilters

Die Namenliste der Abonnementfilter, die mit den ursprünglichen Protokolldaten übereingestimmt haben.

messageType

Datennachrichten verwenden den Typ „DATA_MESSAGE“. Manchmal können CloudWatch Logs Amazon Kinesis Data Streams Streams-Datensätze mit dem Typ „CONTROL_MESSAGE“ ausgeben, hauptsächlich um zu überprüfen, ob das Ziel erreichbar ist.

logEvents

Die tatsächlichen Protokolldaten, die als Array von Protokollereignis-Datensätzen dargestellt werden. Die ID-Eigenschaft ist eine eindeutige Kennung für jedes Protokollereignis.

Ändern der Mitgliedschaft im Ziel zur Laufzeit

In manchen Situationen müssen Sie die Mitgliedschaft einiger Benutzer in einem Ziel, das Ihr Eigentum ist, hinzufügen oder entfernen. Sie können den Befehl `put-destination-policy` für Ihr Ziel mit einer neuen Zugriffsrichtlinie verwenden. Im folgenden Beispiel wird festgelegt, dass das zuvor hinzugefügte Konto 111111111111 keine Protokolldaten mehr sendet und das Konto 222222222222 aktiviert wird.

1. Rufen Sie die Richtlinie ab, die derzeit mit dem Ziel `TestDestination` verknüpft ist, und notieren Sie sich Folgendes: `AccessPolicy`

```
aws logs describe-destinations \
  --destination-name-prefix "testDestination"

{
  "Destinations": [
    {
      "DestinationName": "testDestination",
      "RoleArn": "arn:aws:iam::999999999999:role/CWLtoKinesisRole",
      "DestinationArn":
        "arn:aws:logs:region:999999999999:destination:testDestination",
      "TargetArn": "arn:aws:kinesis:region:999999999999:stream/RecipientStream",
      "AccessPolicy": "{\n\"Version\": \"2012-10-17\", \"Statement\":
        [\n{\n\"Sid\": \"\", \"Effect\": \"Allow\", \"Principal\": {\n\"AWS\":
        \n\"111111111111\"}, \"Action\": \"logs:PutSubscriptionFilter\", \"Resource\":
        \n\"arn:aws:logs:region:999999999999:destination:testDestination\"}] }"
```

```
]
}
```

2. Aktualisieren Sie die Richtlinie, sodass angezeigt wird, dass das Konto 111111111111 angehalten wurde und das Konto 222222222222 aktiviert wurde. Fügen Sie diese Richtlinie in die Datei ~/NewAccessPolicy.json ein:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "222222222222"
      },
      "Action": "logs:PutSubscriptionFilter",
      "Resource": "arn:aws:logs:us-  
east-1:999999999999:destination:testDestination"
    }
  ]
}
```

3. Rufen Sie PutDestinationPolicy auf, um die in der NewAccessPolicy.json-Datei definierte Richtlinie dem Ziel zuzuordnen:

```
aws logs put-destination-policy \
--destination-name "testDestination" \
--access-policy file://~/NewAccessPolicy.json
```

Damit werden schließlich die Protokollereignisse von der Konto-ID 111111111111 deaktiviert. Protokollereignisse der Konto-ID 222222222222 werden an das Ziel übertragen, sobald der Eigentümer des Kontos 222222222222 einen Abonnementfilter erstellt.

Aktualisieren eines bestehenden kontoübergreifenden Abonnements

Wenn Sie derzeit über ein kontoübergreifendes Protokoll-Abonnement verfügen, bei dem das Zielkonto Berechtigungen nur bestimmten Senderkonten gewährt, und Sie dieses Abonnement

aktualisieren möchten, damit das Zielkonto Zugriff auf alle Konten in einer Organisation gewährt, führen Sie die Schritte in diesem Abschnitt aus.

Themen

- [Schritt 1: Aktualisieren der Abonnementfilter](#)
- [Schritt 2: Aktualisieren der bestehenden Richtlinie für die Zugriffsrichtlinie](#)

Schritt 1: Aktualisieren der Abonnementfilter

Note

Dieser Schritt ist nur für kontenübergreifende Abonnements für Protokolle erforderlich, die von den in [Protokollierung aktivieren von AWS service](#) aufgeführten Services erstellt werden. Wenn Sie nicht mit Protokollen arbeiten, die von einer dieser Protokollgruppen erstellt wurden, können Sie zu [Schritt 2: Aktualisieren der bestehenden Richtlinie für die Zugriffsrichtlinie](#) wechseln.

In bestimmten Fällen müssen Sie die Abonnementfilter in allen Absenderkonten aktualisieren, die Protokolle an das Zielkonto senden. Das Update fügt eine IAM-Rolle hinzu, die davon ausgehen und überprüfen CloudWatch kann, dass das Absenderkonto berechtigt ist, Protokolle an das Empfängerkonto zu senden.

Befolgen Sie die Schritte in diesem Abschnitt für jedes Senderkonto, das Sie aktualisieren möchten, um die Organisations-ID für die kontoübergreifenden Abonnementberechtigungen zu verwenden.

In den Beispielen in diesem Abschnitt wurden für zwei Konten, 111111111111 und 222222222222, bereits Abonnementfilter erstellt, um Protokolle an Konto 999999999999 zu senden. Die vorhandenen Abonnementfilterwerte lauten wie folgt:

```
## Existing Subscription Filter parameter values
\ --log-group-name "my-log-group-name"
\ --filter-name "RecipientStream"
\ --filter-pattern "${$.userIdentity.type = Root}"
\ --destination-arn "arn:aws:logs:region:999999999999:destination:testDestination"
```

Wenn Sie die aktuellen Parameterwerte des Abonnementfilters suchen müssen, geben Sie den folgenden Befehl ein.

```
aws logs describe-subscription-filters
\ --log-group-name "my-log-group-name"
```

Um einen Abonnementfilter zu aktualisieren, um ab sofort Organisation IDs für kontoübergreifende Protokollberechtigungen zu verwenden

1. Erstellen Sie die folgende Vertrauensrichtlinie in einer Datei namens `~/TrustPolicyForCWL.json`. Erstellen Sie in einem Text-Editor die Richtliniendatei, verwenden Sie nicht die IAM-Konsole.

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

2. Erstellen Sie als Nächstes die IAM-Rolle, die diese Richtlinie verwendet. Notieren Sie sich den Arn-Wert des Arn-Werts, der vom Befehl zurückgegeben wird, den Sie später in diesem Verfahren benötigen. In diesem Beispiel verwenden wir `CWLtoSubscriptionFilterRole` für den Namen der Rolle, die wir erstellen.

```
aws iam create-role
\ --role-name CWLtoSubscriptionFilterRole
\ --assume-role-policy-document file://~/TrustPolicyForCWL.json
```

3. Erstellen Sie eine Berechtigungsrichtlinie, um die Aktionen zu definieren, die CloudWatch Logs auf Ihrem Konto ausführen kann.
 - a. Verwenden Sie zunächst einen Text-Editor, um die folgende Berechtigungsrichtlinie in einer Datei mit dem Namen zu erstellen `/PermissionsForCWLSubscriptionFilter.json`.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:PutLogEvents",
      "Resource": "arn:aws:logs:region:111111111111:log-
group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
```

```
    }
  ]
}
```

- b. Geben Sie den folgenden Befehl ein, um die soeben erstellte Berechtigungsrichtlinie mit der Rolle zu verknüpfen, die Sie in Schritt 2 erstellt haben.

```
aws iam put-role-policy
  --role-name CWLtoSubscriptionFilterRole
  --policy-name Permissions-Policy-For-CWL-Subscription-filter
  --policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

4. Geben Sie den folgenden Befehl ein, um den Abonnementfilter zu aktualisieren.

```
aws logs put-subscription-filter
  \ --log-group-name "my-log-group-name"
  \ --filter-name "RecipientStream"
  \ --filter-pattern "${.userIdentity.type = Root}"
  \ --destination-arn
  "arn:aws:logs:region:999999999999:destination:testDestination"
  \ --role-arn "arn:aws:iam::111111111111:role/CWLtoSubscriptionFilterRole"
```

Schritt 2: Aktualisieren der bestehenden Richtlinie für die Zugriffsrichtlinie

Nachdem Sie die Abonnementfilter in allen Senderkonten aktualisiert haben, können Sie die Zielzugriffsrichtlinie im Empfängerkonto aktualisieren.

In den folgenden Beispielen ist das Empfängerkonto 999999999999 und das Ziel hat den Namen testDestination.

Das Update ermöglicht es allen Konten, die Teil der Organisation mit der ID o-1234567890 sind, Protokolle an das Empfängerkonto zu senden. Nur die Konten, für die Abonnementfilter erstellt wurden, senden tatsächlich Protokolle an das Empfängerkonto.

So aktualisieren Sie die Zielzugriffsrichtlinie im Empfängerkonto, um eine Organisations-ID für Berechtigungen zu verwenden

1. Verwenden Sie im Empfängerkonto einen Texteditor, um eine ~/AccessPolicy.json-Datei mit dem folgenden Inhalt zu erstellen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "logs:PutSubscriptionFilter",
      "Resource": "arn:aws:logs:us-east-1:999999999999:destination:testDestination",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": [
            "o-1234567890"
          ]
        }
      }
    }
  ]
}
```

2. Geben Sie den folgenden Befehl ein, um die Richtlinie, die Sie soeben erstellt haben, dem vorhandenen Ziel anzufügen. Um ein Ziel so zu aktualisieren, dass statt einer Zugriffsrichtlinie, die ein bestimmtes AWS Konto auflistet, eine Zugriffsrichtlinie mit einer Organisations-ID verwendet wird IDs, fügen Sie den `force` Parameter hinzu.

 Warning

Wenn Sie mit Protokollen arbeiten, die von einem AWS Dienst gesendet wurden, der unter aufgeführt ist [Protokollierung aktivieren von AWS service](#), müssen Sie, bevor Sie diesen Schritt ausführen, zuerst die Abonnementfilter in allen Absenderkonten aktualisiert haben, wie unter beschrieben [Schritt 1: Aktualisieren der Abonnementfilter](#).

```
aws logs put-destination-policy
\ --destination-name "testDestination"
\ --access-policy file://~/AccessPolicy.json
```

```
\ --force
```

Kontoübergreifender regionsübergreifender Austausch von Protokolldaten mit Firehose

Wenn Sie Protokolldaten für mehrere Konten freigeben möchten, müssen Sie Sender und Receiver der Protokolldaten festlegen:

- **Absender der Protokolldaten** — Ruft die Zielinformationen vom Empfänger ab und teilt Logs mit, dass CloudWatch Logs bereit ist, seine Protokollereignisse an das angegebene Ziel zu senden. In den Verfahren im Rest dieses Abschnitts wird der Absender der Protokolldaten mit der fiktiven AWS Kontonummer 1111111111 angezeigt.
- **Empfänger von Protokolldaten** — richtet ein Ziel ein, das einen Amazon Kinesis Data Streams Streams-Stream kapselt und CloudWatch Logs darüber informiert, dass der Empfänger Protokolldaten empfangen möchte. Der Empfänger gibt dann die Informationen über sein Ziel für den Absender frei. In den Verfahren im Rest dieses Abschnitts wird der Empfänger der Protokolldaten mit der fiktiven AWS Kontonummer 222222222222 angezeigt.

Das Beispiel in diesem Abschnitt verwendet einen Firehose-Lieferstream mit Amazon S3 S3-Speicher. Sie können Firehose-Lieferstreams auch mit unterschiedlichen Einstellungen einrichten. Weitere Informationen finden Sie unter [Firehose Delivery Stream erstellen](#).

Note

Die Protokollgruppe und das Ziel müssen sich in derselben AWS Region befinden. Die AWS Ressource, auf die das Ziel verweist, kann sich jedoch in einer anderen Region befinden.

Note

Der Firehose-Abonnementfilter für dasselbe Konto und denselben regionsübergreifenden Lieferstream wird unterstützt.

Themen

- [Schritt 1: Erstellen Sie einen Firehose-Lieferstream](#)

- [Schritt 2: Erstellen eines Ziels](#)
- [Schritt 3: Add/validate IAM-Berechtigungen für das kontoübergreifende Ziel](#)
- [Schritt 4: Erstellen eines Abonnementfilters](#)
- [Validierung des Flusses von Protokollereignissen](#)
- [Ändern der Mitgliedschaft im Ziel zur Laufzeit](#)

Schritt 1: Erstellen Sie einen Firehose-Lieferstream

Important

Bevor Sie die folgenden Schritte ausführen, müssen Sie eine Zugriffsrichtlinie verwenden, damit Firehose auf Ihren Amazon S3 S3-Bucket zugreifen kann. Weitere Informationen finden Sie unter [Zugriffskontrolle](#) im Amazon Data Firehose Developer Guide.

Alle Schritte in diesem Abschnitt (Schritt 1) müssen im Konto des Protokolldatenempfängers ausgeführt werden.

USA Ost (Nord-Virginia) wird in den folgenden Beispielbefehlen verwendet. Ersetzen Sie diese Region durch die richtige Region für Ihre Bereitstellung.

Um einen Firehose-Lieferstream zu erstellen, der als Ziel verwendet werden soll

1. Erstellen eines Amazon-S3-Buckets:

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --create-bucket-configuration
LocationConstraint=us-east-1
```

2. Erstellen Sie die IAM-Rolle, die Firehose die Berechtigung erteilt, Daten in den Bucket zu legen.

- a. Verwenden Sie zunächst einen Text-Editor zum Erstellen einer Vertrauensrichtlinie in einer Datei `~/TrustPolicyForFirehose.json`.

```
{ "Statement": { "Effect": "Allow", "Principal": { "Service":
  "firehose.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition":
  { "StringEquals": { "sts:ExternalId":"222222222222" } } } }
```

- b. Erstellen Sie die IAM-Rolle und geben Sie die Datei mit der Vertrauensrichtlinie an, die Sie gerade erstellt haben.

```
aws iam create-role \
  --role-name FirehoseToS3Role \
  --assume-role-policy-document file://~/TrustPolicyForFirehose.json
```

- c. Die Ausgabe für den Befehl sieht dann wie folgt aus. Notieren Sie sich den Rollennamen und die Rollen-ARN.

```
{
  "Role": {
    "Path": "/",
    "RoleName": "FirehoseToS3Role",
    "RoleId": "AR0AR3BXASEKW7K635M53",
    "Arn": "arn:aws:iam::222222222222:role/FirehoseToS3Role",
    "CreateDate": "2021-02-02T07:53:10+00:00",
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Effect": "Allow",
        "Principal": {
          "Service": "firehose.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
          "StringEquals": {
            "sts:ExternalId": "222222222222"
          }
        }
      }
    }
  }
}
```

3. Erstellen Sie eine Berechtigungsrichtlinie, um die Aktionen zu definieren, die Firehose in Ihrem Konto ausführen kann.
- a. Verwenden Sie zunächst einen Text-Editor, um die folgende Berechtigungsrichtlinie in einer Datei mit dem Namen zu erstellen `~/PermissionsForFirehose.json`. Abhängig von Ihrem Anwendungsfall müssen Sie dieser Datei möglicherweise weitere Berechtigungen hinzufügen.

```
{
  "Statement": [{
```

```

    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl",
        "s3:ListBucket"
    ],
    "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
    ]
  }]
}
```

- b. Geben Sie den folgenden Befehl ein, um die soeben erstellte Berechtigungsrichtlinie der IAM-Rolle zuzuordnen.

```

aws iam put-role-policy --role-name FirehoseToS3Role --policy-name
Permissions-Policy-For-Firehose-To-S3 --policy-document file://~/
PermissionsForFirehose.json
```

4. Geben Sie den folgenden Befehl ein, um den Firehose-Lieferstream zu erstellen. Ersetzen Sie *my-role-arn* und *amzn-s3-demo-bucket2-arn* durch die richtigen Werte für Ihre Bereitstellung.

```

aws firehose create-delivery-stream \
  --delivery-stream-name 'my-delivery-stream' \
  --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::222222222222:role/FirehoseToS3Role", "BucketARN":
  "arn:aws:s3:::amzn-s3-demo-bucket"}'
```

Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```

{
  "DeliveryStreamARN": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream"
}
```

Schritt 2: Erstellen eines Ziels

Important

Alle Schritte in diesem Verfahren müssen im Konto des Protokolldatenempfängers ausgeführt werden.

Wenn das Ziel erstellt wurde, sendet CloudWatch Logs im Namen des Empfängerkontos eine Testnachricht an das Ziel. Wenn der Abonnementfilter später aktiv ist, sendet CloudWatch Logs im Namen des Quellkontos Protokollereignisse an das Ziel.

So erstellen Sie ein Ziel

1. Warten Sie, bis der Firehose-Stream, in dem Sie ihn erstellt haben, aktiv [Schritt 1: Erstellen Sie einen Firehose-Lieferstream](#) wird. Sie können den folgenden Befehl verwenden, um das StreamDescription zu überprüfen. StreamStatusEigentum.

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
```

Beachten Sie außerdem die DeliveryStreamDescription. DeliveryStreamARN-Wert, da Sie ihn in einem späteren Schritt verwenden müssen. Beispielausgabe dieses Befehls:

```
{
  "DeliveryStreamDescription": {
    "DeliveryStreamName": "my-delivery-stream",
    "DeliveryStreamARN": "arn:aws:firehose:us-east-1:222222222222:deliverystream/my-delivery-stream",
    "DeliveryStreamStatus": "ACTIVE",
    "DeliveryStreamEncryptionConfiguration": {
      "Status": "DISABLED"
    },
    "DeliveryStreamType": "DirectPut",
    "VersionId": "1",
    "CreateTimestamp": "2021-02-01T23:59:15.567000-08:00",
    "Destinations": [
      {
        "DestinationId": "destinationId-000000000001",
        "S3DestinationDescription": {
          "RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
```

```

        "BufferingHints": {
            "SizeInMBs": 5,
            "IntervalInSeconds": 300
        },
        "CompressionFormat": "UNCOMPRESSED",
        "EncryptionConfiguration": {
            "NoEncryptionConfig": "NoEncryption"
        },
        "CloudWatchLoggingOptions": {
            "Enabled": false
        }
    },
    "ExtendedS3DestinationDescription": {
        "RoleARN": "arn:aws:iam::222222222222:role/FirehoseToS3Role",
        "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
        "BufferingHints": {
            "SizeInMBs": 5,
            "IntervalInSeconds": 300
        },
        "CompressionFormat": "UNCOMPRESSED",
        "EncryptionConfiguration": {
            "NoEncryptionConfig": "NoEncryption"
        },
        "CloudWatchLoggingOptions": {
            "Enabled": false
        },
        "S3BackupMode": "Disabled"
    }
},
],
"HasMoreDestinations": false
}

```

Es kann einige Minuten dauern, bis der Bereitstellungsdatenstrom im aktiven Status angezeigt wird.

2. Wenn der Lieferstream aktiv ist, erstellen Sie die IAM-Rolle, die CloudWatch Logs die Erlaubnis erteilt, Daten in Ihren Firehose-Stream einzufügen. Zunächst müssen Sie eine Vertrauensrichtlinie in einer Datei `TrustPolicyFor~/.CWL.json` erstellen. Verwenden Sie einen Text-Editor, um diese Richtlinie zu erstellen. Weitere Informationen zu CloudWatch Logs-Endpunkten finden Sie unter [Amazon CloudWatch Logs-Endpunkte und Kontingente](#).

Diese Richtlinie enthält einen globalen Bedingungskontextschlüssel `aws:SourceArn`, der das `sourceAccountId` angibt, um das Confused-Deputy-Problem zu vermeiden. Wenn Sie die Quell-Kontonummer beim ersten Aufruf noch nicht kennen, empfehlen wir Ihnen, die Ziel-ARN in das Quell-ARN-Feld einzutragen. Bei den folgenden Aufrufen sollten Sie als Quell-ARN den tatsächlichen Quell-ARN angeben, den Sie beim ersten Aufruf ermittelt haben. Weitere Informationen finden Sie unter [Confused-Deputy-Prävention](#).

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "Service": "logs.region.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringLike": {
        "aws:SourceArn": [
          "arn:aws:logs:region:sourceAccountId:",
          "arn:aws:logs:region:recipientAccountId:"
        ]
      }
    }
  }
}
```

3. Verwenden Sie den Befehl `aws iam create-role`, um die IAM-Rolle zu erstellen und geben Sie die Datei mit der Vertrauensrichtlinie an, die Sie gerade erstellt haben.

```
aws iam create-role \
  --role-name CWLtoKinesisFirehoseRole \
  --assume-role-policy-document file://~/TrustPolicyForCWL.json
```

Dies ist eine Beispielausgabe. Notieren Sie den zurückgegebenen `Role.Arn`-Wert, da Sie ihn in einem späteren Schritt benötigen.

```
{
  "Role": {
    "Path": "/",
    "RoleName": "CWLtoKinesisFirehoseRole",
    "RoleId": "AR0AR3BXASEKYJYWF243H",
```

```

    "Arn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
    "CreateDate": "2021-02-02T08:10:43+00:00",
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Effect": "Allow",
        "Principal": {
          "Service": "logs.region.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
          "StringLike": {
            "aws:SourceArn": [
              "arn:aws:logs:region:sourceAccountId:*",
              "arn:aws:logs:region:recipientAccountId:"
            ]
          }
        }
      }
    }
  }
}

```

- Erstellen Sie eine Berechtigungsrichtlinie, um zu definieren, welche Aktionen CloudWatch Logs auf Ihrem Konto ausführen kann. Verwenden Sie zunächst einen Texteditor, um eine Berechtigungsrichtlinie in einer Datei ~/PermissionsFor CWL.json zu erstellen:

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["firehose:*"],
      "Resource": ["arn:aws:firehose:region:222222222222:*"]
    }
  ]
}

```

- Verknüpfen Sie mit dem folgenden Befehl die Berechtigungsrichtlinie mit der Rolle:

```

aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-name
Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json

```

- Nachdem sich der Firehose-Lieferstream im aktiven Zustand befindet und Sie die IAM-Rolle erstellt haben, können Sie das CloudWatch Logs-Ziel erstellen.

- a. In diesem Schritt wird keine Zugriffsrichtlinie mit Ihrem Ziel verknüpft. Es ist zudem erst der erste von zwei Schritten zum Erstellen eines Ziels. Notieren Sie sich die in der Nutzlast zurückgegebene ARN des neuen Zielortes, da Sie diese in einem späteren Schritt als `destination.arn` verwenden werden.

```
aws logs put-destination \  
  
    --destination-name "testFirehoseDestination" \  
    --target-arn "arn:aws:firehose:us-east-1:222222222222:deliverystream/my-delivery-stream" \  
    --role-arn "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole"  
  
{  
    "destination": {  
        "destinationName": "testFirehoseDestination",  
        "targetArn": "arn:aws:firehose:us-east-1:222222222222:deliverystream/my-delivery-stream",  
        "roleArn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",  
        "arn": "arn:aws:logs:us-east-1:222222222222:destination:testFirehoseDestination"}  
}
```

- b. Verknüpfen Sie, nachdem der vorherige Schritt abgeschlossen ist, im Empfängerkonto der Protokolldaten (222222222222) eine Zugriffsrichtlinie mit dem Ziel.

Diese Richtlinie ermöglicht dem Sender-Konto der Protokolldaten (111111111111), auf das Ziel nur im Empfängerkonto der Protokolldaten (222222222222) zuzugreifen. Sie können einen Texteditor verwenden, um diese Richtlinie in die Datei `~/AccessPolicy.json` einzufügen:

JSON

```
{  
    "Version": "2012-10-17",  
    "Statement" : [  
        {  
            "Sid" : "",  
            "Effect" : "Allow",  
            "Principal" : {  
                "AWS" : "111111111111"
```



```
    },  
    "Action" : "logs:PutSubscriptionFilter",  
    "Resource" : "arn:aws:logs:us-  
east-1:222222222222:destination:testFirehoseDestination"  
  }  
]  
}
```

- c. Damit wird eine Richtlinie erstellt, die bestimmt, wer Schreibzugriff auf das Ziel hat. In dieser Richtlinie muss die PutSubscriptionFilter Aktion logs: für den Zugriff auf das Ziel angegeben werden. Kontoübergreifende Benutzer verwenden die Aktion PutSubscriptionFilter, um Protokollereignisse an das Ziel zu senden:

```
aws logs put-destination-policy \  
  --destination-name "testFirehoseDestination" \  
  --access-policy file://~/AccessPolicy.json
```

Schritt 3: Add/validate IAM-Berechtigungen für das kontoübergreifende Ziel

Gemäß der Bewertungslogik für AWS kontenübergreifende Richtlinien muss für den Zugriff auf eine kontoübergreifende Ressource (z. B. ein Kinesis- oder Firehose-Stream, der als Ziel für einen Abonnementfilter verwendet wird) eine identitätsbasierte Richtlinie im sendenden Konto vorhanden sein, die expliziten Zugriff auf die kontenübergreifende Zielressource gewährt. Weitere Informationen zur Logik der Richtlinienbewertung finden Sie unter [Bewertungslogik für kontenübergreifende Richtlinien](#).

Sie können die identitätsbasierte Richtlinie der IAM-Rolle oder dem IAM-Benutzer zuordnen, die Sie zum Erstellen des Abonnementfilters verwenden. Diese Richtlinie muss im übermittelnden Konto vorhanden sein. Wenn Sie die Administratorrolle verwenden, um den Abonnementfilter zu erstellen, können Sie diesen Schritt überspringen und mit [Schritt 4: Erstellen eines Abonnementfilters](#) fortfahren.

Um die IAM-Berechtigungen hinzuzufügen oder zu validieren, die für kontoübergreifende Konten erforderlich sind

1. Geben Sie den folgenden Befehl ein, um zu überprüfen, welche IAM-Rolle oder welcher IAM-Benutzer zur Ausführung von Log-Befehlen verwendet wird. AWS

```
aws sts get-caller-identity
```

Daraufhin erhalten Sie ein Ergebnis, das dem hier dargestellten entspricht:

```
{
  "UserId": "User ID",
  "Account": "sending account id",
  "Arn": "arn:aws:sending account id:role/user:RoleName/UserName"
}
```

Notieren Sie sich den Wert, der durch *RoleName* oder dargestellt wird. *UserName*

2. Melden Sie sich AWS-Managementkonsole im sendenden Konto an und suchen Sie nach den angehängten Richtlinien mit der IAM-Rolle oder dem IAM-Benutzer, die in der Ausgabe des in Schritt 1 eingegebenen Befehls zurückgegeben wurden.
3. Stellen Sie sicher, dass die mit dieser Rolle oder diesem Benutzer verknüpften Richtlinien explizite Zugriffsberechtigungen für die `logs:PutSubscriptionFilter` auf der kontoübergreifenden Zielressource enthalten.

Die folgende Richtlinie gewährt Berechtigungen zum Erstellen eines Abonnementfilters für jede Zielressource nur in einem einzigen AWS Konto, einem Konto: 999999999999

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid":
      "AllowSubscriptionFiltersOnAnyResourceInOneSpecificAccount",
      "Effect": "Allow",
      "Action": "logs:PutSubscriptionFilter",
      "Resource": [
        "arn:aws:logs::*:log-group:*",
        "arn:aws:logs*:123456789012:destination:*"
      ]
    }
  ]
}
```

Die folgende Richtlinie gewährt Berechtigungen zum Erstellen eines Abonnementfilters nur für eine bestimmte Zielressource mit `sampleDestination` dem Namen „AWS Einzelkonto“123456789012:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSubscriptionFiltersOnSpecificResource",
      "Effect": "Allow",
      "Action": "logs:PutSubscriptionFilter",
      "Resource": [
        "arn:aws:logs:*:*:log-group:*",
        "arn:aws:logs:*:123456789012:destination:amzn-s3-demo-bucket"
      ]
    }
  ]
}
```

Schritt 4: Erstellen eines Abonnementfilters

Wechseln Sie zum Senderkonto, das in diesem Beispiel 111111111111 lautet. Sie erstellen nun den Abonnementfilter im Senderkonto. In diesem Beispiel ist der Filter mit einer Protokollgruppe verknüpft, die AWS CloudTrail Ereignisse enthält, sodass jede protokollierte Aktivität, die mit „Root“ AWS -Anmeldeinformationen ausgeführt wird, an das Ziel übermittelt wird, das Sie zuvor erstellt haben. Weitere Informationen zum Senden von AWS CloudTrail Ereignissen an CloudWatch Protokolle finden Sie unter [Senden von CloudTrail Ereignissen an CloudWatch Protokolle](#) im AWS CloudTrail Benutzerhandbuch.

Wenn Sie den folgenden Befehl eingeben, stellen Sie sicher, dass Sie als IAM-Benutzer angemeldet sind oder die IAM-Rolle verwenden, für die Sie die Richtlinie hinzugefügt haben, in [Schritt 3: Add/validate IAM-Berechtigungen für das kontoübergreifende Ziel](#).

```
aws logs put-subscription-filter \
  --log-group-name "aws-cloudtrail-logs-111111111111-300a971e" \
  --filter-name "firehose_test" \
```

```
--filter-pattern "${.userIdentity.type = AssumedRole}" \
--destination-arn "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
```

Die Protokollgruppe und das Ziel müssen sich in derselben AWS Region befinden. Das Ziel kann jedoch auf eine AWS Ressource verweisen, z. B. auf einen Firehose-Stream, der sich in einer anderen Region befindet.

Validierung des Flusses von Protokollereignissen

Nachdem Sie den Abonnementfilter erstellt haben, leitet CloudWatch Logs alle eingehenden Protokollereignisse, die dem Filtermuster entsprechen, an den Firehose-Lieferstream weiter. Die Daten werden in Ihrem Amazon S3 S3-Bucket basierend auf dem Zeitpufferintervall angezeigt, das für den Firehose-Lieferstream festgelegt ist. Sobald genügend Zeit abgelaufen ist, können Sie die Daten im Amazon-S3-Bucket überprüfen. Geben Sie zur Prüfung des Buckets den folgenden Befehl ein:

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket'
```

Die Ausgabe des Befehls sieht dann wie folgt aus:

```
{
  "Contents": [
    {
      "Key": "2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba",
      "LastModified": "2021-02-02T09:00:26+00:00",
      "ETag": "\"EXAMPLEa817fb88fc770b81c8f990d\"",
      "Size": 198,
      "StorageClass": "STANDARD",
      "Owner": {
        "DisplayName": "firehose+2test",
        "ID": "EXAMPLE27fd05889c665d2636218451970ef79400e3d2aecca3adb1930042e0"
      }
    }
  ]
}
```

Sie können dann ein bestimmtes Objekt aus dem Bucket abrufen, indem Sie den folgenden Befehl eingeben. Ersetzen Sie den Wert von key mit dem Wert, den Sie im vorherigen Befehl gefunden haben.

```
aws s3api get-object --bucket 'amzn-s3-demo-bucket' --key '2021/02/02/08/my-delivery-stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba' testfile.gz
```

Die Daten im Amazon-S3-Objekt sind im GZIP-Format komprimiert. Sie können die Rohdaten über die Befehlszeile mit einem der folgenden Befehlen überprüfen:

Linux:

```
zcat testfile.gz
```

macOS:

```
zcat <testfile.gz
```

Ändern der Mitgliedschaft im Ziel zur Laufzeit

In manchen Situationen müssen Sie Protokollsender in einem Ziel, das Ihr Eigentum ist, hinzufügen oder entfernen. Sie können die PutDestinationPolicyAktion auf Ihrem Ziel mit einer neuen Zugriffsrichtlinie verwenden. Im folgenden Beispiel wird festgelegt, dass das zuvor hinzugefügte Konto 111111111111 keine Protokolldaten mehr senden kann und das Konto 333333333333 aktiviert wird.

1. Rufen Sie die Richtlinie ab, die derzeit mit dem Ziel TestDestination verknüpft ist, und notieren Sie sich Folgendes: AccessPolicy

```
aws logs describe-destinations \
  --destination-name-prefix "testFirehoseDestination"

{
  "destinations": [
    {
      "destinationName": "testFirehoseDestination",
      "targetArn": "arn:aws:firehose:us-east-1:222222222222:deliverystream/my-delivery-stream",
      "roleArn": "arn:aws:iam:: 222222222222:role/CWLtoKinesisFirehoseRole",
      "accessPolicy": "{\n  \"Version\" : \"2012-10-17\",\n  \"Statement\" : [\n    {\n      \"Sid\" : \"\",\n      \"Effect\" : \"Allow\",\n      \"Principal\" : {\n        \"AWS\" : \"111111111111\"\n      },\n      \"Action\" : \"logs:PutSubscriptionFilter\",\n      \"Resource\" : \"arn:aws:logs:us-east-1:222222222222:destination:testFirehoseDestination\"\n    }\n  ]\n}"
```

```

        "arn": "arn:aws:logs:us-east-1:
222222222222:destination:testFirehoseDestination",
        "creationTime": 1612256124430
    }
]
}

```

2. Aktualisieren Sie die Richtlinie dementsprechend, dass das Konto 111111111111 angehalten wurde und das Konto 333333333333 aktiviert wurde. Fügen Sie diese Richtlinie in die Datei `~/NewAccessPolicy.json` ein:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "333333333333"
      },
      "Action": "logs:PutSubscriptionFilter",
      "Resource": "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}

```

3. Verwenden Sie den folgenden Befehl, um die in der `NewAccessPolicyJSON`-Datei definierte Richtlinie dem Ziel zuzuordnen:

```

aws logs put-destination-policy \
  --destination-name "testFirehoseDestination" \

  --access-policy file://~/NewAccessPolicy.json

```

Damit werden schließlich die Protokollereignisse von der Konto-ID 111111111111 deaktiviert. Protokollereignisse der Konto-ID 333333333333 werden an das Ziel übertragen, sobald der Eigentümer des Kontos 333333333333 einen Abonnementfilter erstellt.

Kontoübergreifende, regionsübergreifende Abonnements auf Kontoebene mit Amazon Kinesis Data Streams

Wenn Sie ein kontoübergreifendes Abonnement erstellen, können Sie ein einzelnes Konto oder eine Organisation angeben, die der Sender sein soll. Wenn Sie eine Organisation angeben, können alle Konten in der Organisation mit diesem Verfahren Protokolle an das Receiver-Konto senden.

Wenn Sie Protokolldaten für mehrere Konten freigeben möchten, müssen Sie Sender und Receiver der Protokolldaten festlegen:

- **Absender der Protokolldaten** — Ruft die Zielinformationen vom Empfänger ab und teilt Logs mit, dass CloudWatch Logs bereit ist, seine Protokollereignisse an das angegebene Ziel zu senden. In den Verfahren im Rest dieses Abschnitts wird der Absender der Protokolldaten mit der fiktiven AWS Kontonummer 1111111111 angezeigt.

Wenn mehrere Konten in einer Organisation Protokolle an ein Empfängerkonto senden, können Sie eine Richtlinie erstellen, die allen Konten in der Organisation die Berechtigung zum Senden von Protokollen an das Empfängerkonto gewährt. Sie müssen immer noch separate Abonnementfilter für jedes Senderkonto einrichten.

- **Empfänger von Protokolldaten** — richtet ein Ziel ein, das einen Amazon Kinesis Data Streams Streams-Stream kapselt und CloudWatch Logs darüber informiert, dass der Empfänger Protokolldaten empfangen möchte. Der Empfänger gibt dann die Informationen über sein Ziel für den Absender frei. In den Verfahren im Rest dieses Abschnitts wird der Empfänger der Protokolldaten mit der fiktiven AWS Kontonummer 999999999999 angezeigt.

Um Protokollereignisse von kontoübergreifenden Benutzern zu empfangen, erstellt der Empfänger der Protokolldaten zunächst ein Protokollziel. CloudWatch Jedes Ziel umfasst die folgenden Schlüsselemente:

Name des Ziels

Der Name der Zielregion, die Sie erstellen möchten.

Ziel-ARN

Der Amazon-Ressourcenname (ARN) der AWS Ressource, die Sie als Ziel für den Abonnement-Feed verwenden möchten.

ARN der Rolle

Eine AWS Identity and Access Management (IAM) -Rolle, die CloudWatch Logs die erforderlichen Berechtigungen erteilt, um Daten in den ausgewählten Stream zu übertragen.

Zugriffsrichtlinie

Ein IAM-Richtliniendokument (im JSON-Format, das in der IAM-Richtliniengrammatik geschrieben ist), das kontrolliert, welche Benutzer Schreibberechtigung für das Ziel haben.

Note

Die Protokollgruppe und das Ziel müssen sich in derselben AWS Region befinden. Die AWS Ressource, auf die das Ziel verweist, kann sich jedoch in einer anderen Region befinden. In den Beispielen in den folgenden Abschnitten werden alle regionsspezifischen Ressourcen in USA Ost (Nord-Virginia) erstellt.

Themen

- [Einrichten eines neuen kontoübergreifenden Abonnements](#)
- [Aktualisieren eines bestehenden kontoübergreifenden Abonnements](#)

Einrichten eines neuen kontoübergreifenden Abonnements

Befolgen Sie die Schritte in diesen Abschnitten, um ein neues kontoübergreifendes Protokollabonnement einzurichten.

Themen

- [Schritt 1: Erstellen eines Ziels](#)
- [Schritt 2: \(Nur bei Verwendung einer Organisation\) Erstellen Sie eine IAM-Rolle](#)
- [Schritt 3: Erstellen Sie eine Abonnementfilterrichtlinie auf Kontoebene](#)
- [Validierung des Ablaufs von Protokollereignissen](#)
- [Ändern der Mitgliedschaft im Ziel zur Laufzeit](#)

Schritt 1: Erstellen eines Ziels

Important

Alle Schritte in diesem Verfahren müssen im Konto des Protokolldatenempfängers ausgeführt werden.

In diesem Beispiel hat das Konto des Empfängers der Protokolldaten die Konto-ID 9999999999, während die AWS Konto-ID des Absenders der Protokolldaten 1111111111 lautet. AWS

In diesem Beispiel wird ein Ziel mithilfe eines Amazon Kinesis Data Streams-Streams namens und einer Rolle erstellt RecipientStream, die es CloudWatch Logs ermöglicht, Daten darauf zu schreiben.

Wenn das Ziel erstellt ist, sendet CloudWatch Logs im Namen des Empfängerkontos eine Testnachricht an das Ziel. Wenn der Abonnementfilter später aktiv ist, sendet CloudWatch Logs im Namen des Quellkontos Protokollereignisse an das Ziel.

So erstellen Sie ein Ziel

1. Erstellen Sie im Empfängerkonto einen Ziel-Stream in Amazon Kinesis Data Streams. Geben Sie an der Eingabeaufforderung Folgendes ein:

```
aws kinesis create-stream --stream-name "RecipientStream" --shard-count 1
```

2. Warten Sie, bis der -Stream aktiv wird. Sie können den Befehl `aws kinesis describe-stream` verwenden, um das zu überprüfen. StreamDescription StreamStatusEigentum. Notieren Sie sich außerdem den Wert StreamDescription.streamArn, da Sie ihn später an Logs übergeben werden: CloudWatch

```
aws kinesis describe-stream --stream-name "RecipientStream"
{
  "StreamDescription": {
    "StreamStatus": "ACTIVE",
    "StreamName": "RecipientStream",
    "StreamARN": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "34028236692093846346337460743176EXAMPLE",
```

```

    "StartingHashKey": "0"
  },
  "SequenceNumberRange": {
    "StartingSequenceNumber":
"4955113521868881845667950383198145878459135270218EXAMPLE"
  }
}
]
}
}

```

Es kann einige Minuten dauern, bis der Stream im aktiven Status angezeigt wird.

3. Erstellen Sie die IAM-Rolle, die CloudWatch Logs die Berechtigung erteilt, Daten in Ihren Stream einzufügen. Zunächst müssen Sie eine Vertrauensrichtlinie in einer Datei `~/TrustPolicyForCWL.json` erstellen. Erstellen Sie in einem Text-Editor die Richtliniendatei, verwenden Sie nicht die IAM-Konsole.

Diese Richtlinie enthält einen globalen Bedingungskontextschlüssel `aws:SourceArn`, der das `sourceAccountId` angibt, um das Confused-Deputy-Problem zu vermeiden. Wenn Sie die Quell-Kontonummer beim ersten Aufruf noch nicht kennen, empfehlen wir Ihnen, die Ziel-ARN in das Quell-ARN-Feld einzutragen. Bei den folgenden Aufrufen sollten Sie als Quell-ARN den tatsächlichen Quell-ARN angeben, den Sie beim ersten Aufruf ermittelt haben. Weitere Informationen finden Sie unter [Confused-Deputy-Prävention](#).

```

{
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "Service": "logs.amazonaws.com"
    },
    "Condition": {
      "StringLike": {
        "aws:SourceArn": [
          "arn:aws:logs:region:sourceAccountId:*",
          "arn:aws:logs:region:recipientAccountId:*"
        ]
      }
    },
    "Action": "sts:AssumeRole"
  }
}

```

```
}
```

4. Verwenden Sie den Befehl `aws iam create-role`, um die IAM-Rolle zu erstellen und die Vertrauensrichtlinie anzugeben. Notieren Sie sich den zurückgegebenen `Role.Arn`-Wert, da er später auch an Logs übergeben wird: CloudWatch

```
aws iam create-role \
--role-name CWLtoKinesisRole \
--assume-role-policy-document file://~/TrustPolicyForCWL.json

{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Condition": {
          "StringLike": {
            "aws:SourceArn": [
              "arn:aws:logs:region:sourceAccountId:*",
              "arn:aws:logs:region:recipientAccountId:*"
            ]
          }
        },
        "Principal": {
          "Service": "logs.amazonaws.com"
        }
      }
    },
    "RoleId": "AA0IIAH450GAB4HC5F431",
    "CreateDate": "2023-05-29T13:46:29.431Z",
    "RoleName": "CWLtoKinesisRole",
    "Path": "/",
    "Arn": "arn:aws:iam::999999999999:role/CWLtoKinesisRole"
  }
}
```

5. Erstellen Sie eine Berechtigungsrichtlinie, um zu definieren, welche Aktionen CloudWatch Logs auf Ihrem Konto ausführen kann. Verwenden Sie zunächst einen Texteditor, um eine Berechtigungsrichtlinie in einer Datei `~/PermissionsForCWL.json` zu erstellen:

```
{
  "Statement": [
```

```
{
  "Effect": "Allow",
  "Action": "kinesis:PutRecord",
  "Resource": "arn:aws:kinesis:region:999999999999:stream/RecipientStream"
}
]
```

6. Ordnen Sie die Berechtigungsrichtlinie der Rolle zu, indem Sie den Befehl `aws iam` verwenden: `put-role-policy`

```
aws iam put-role-policy \
  --role-name CWLtoKinesisRole \
  --policy-name Permissions-Policy-For-CWL \
  --policy-document file:///~/PermissionsForCWL.json
```

7. Nachdem sich der Stream im aktiven Status befindet und Sie die IAM-Rolle erstellt haben, können Sie das CloudWatch Logs-Ziel erstellen.
- a. In diesem Schritt wird keine Zugriffsrichtlinie mit Ihrem Ziel verknüpft. Es ist zudem erst der erste von zwei Schritten zum Erstellen eines Ziels. Notieren Sie sich den in der Nutzlast ausgegebenen Wert `DestinationArn`:

```
aws logs put-destination \
  --destination-name "testDestination" \
  --target-arn "arn:aws:kinesis:region:999999999999:stream/RecipientStream" \
  --role-arn "arn:aws:iam::999999999999:role/CWLtoKinesisRole"

{
  "DestinationName" : "testDestination",
  "RoleArn" : "arn:aws:iam::999999999999:role/CWLtoKinesisRole",
  "DestinationArn" : "arn:aws:logs:us-east-1:999999999999:destination:testDestination",
  "TargetArn" : "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream"
}
```

- b. Verknüpfen Sie, nachdem Schritt 7a abgeschlossen ist, im Empfängerkonto der Protokolldaten eine Zugriffsrichtlinie mit dem Ziel. Diese Richtlinie muss die `PutSubscriptionFilter` Aktion `logs`: spezifizieren und erteilt dem Absenderkonto die Erlaubnis, auf das Ziel zuzugreifen.

Die Richtlinie erteilt dem AWS Konto, das Protokolle sendet, die entsprechenden Berechtigungen. Sie können nur dieses eine Konto in der Richtlinie angeben, oder wenn das Senderkonto Mitglied einer Organisation ist, kann die Richtlinie die Organisations-ID der Organisation angeben. Auf diese Weise können Sie nur eine Richtlinie erstellen, mit der mehrere Konten in einer Organisation Protokolle an dieses Zielkonto senden können.

Verwenden Sie einen Texteditor, um eine Datei mit dem Namen `~/AccessPolicy.json` mit einer der folgenden Richtlinienanweisungen zu erstellen.

Diese erste Beispielrichtlinie ermöglicht es allen Konten in der Organisation, die die ID `o-1234567890` haben, Protokolle an das Empfängerkonto zu senden.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "logs:PutSubscriptionFilter",
        "logs:PutAccountPolicy"
      ],
      "Resource": "arn:aws:logs:us-east-1:999999999999:destination:testDestination",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": [
            "o-1234567890"
          ]
        }
      }
    }
  ]
}
```

In diesem nächsten Beispiel kann nur das Protokolldatenabsenderkonto (111111111111) Protokolle an das Protokolldatenempfängerkonto senden.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": [
        "logs:PutSubscriptionFilter",
        "logs:PutAccountPolicy"
      ],
      "Resource": "arn:aws:logs:us-east-1:999999999999:destination:testDestination"
    }
  ]
}
```

- c. Fügen Sie die Richtlinie, die Sie im vorherigen Schritt erstellt haben, dem Ziel an.

```
aws logs put-destination-policy \
  --destination-name "testDestination" \
  --access-policy file://~/AccessPolicy.json
```

Diese Zugriffsrichtlinie ermöglicht es Benutzern im AWS Konto mit der ID 111111111111, PutSubscriptionFilter gegen das Ziel mit ARN `arn:aws:logs::999999999999:destination:TestDestination` anzurufen. *region* Jeder Versuch PutSubscriptionFilter eines anderen Benutzers, für dieses Ziel anzurufen, wird abgewiesen.

Informationen dazu, wie Sie die Berechtigungen eines Benutzers mit einer Zugriffsrichtlinie prüfen, finden Sie unter [Verwenden der Richtlinienvvalidierung](#) im IAM-Benutzerhandbuch.

Wenn Sie fertig sind und AWS Organizations für Ihre kontoübergreifenden Berechtigungen verwenden, folgen Sie den Schritten unter [Schritt 2: \(Nur bei Verwendung einer Organisation\) Erstellen Sie eine IAM-Rolle](#). Wenn Sie dem anderen Konto Berechtigungen direkt erteilen, anstatt Organizations zu verwenden, können Sie diesen Schritt überspringen und mit fortfahren [Schritt 3: Erstellen Sie eine Abonnementfilterrichtlinie auf Kontoebene](#).

Schritt 2: (Nur bei Verwendung einer Organisation) Erstellen Sie eine IAM-Rolle

Wenn Sie im vorherigen Abschnitt das Ziel mithilfe einer Zugriffsrichtlinie erstellt haben, die der Organisation, in der sich Konto 111111111111 befindet, Berechtigungen erteilt, anstatt Konto 111111111111 direkt Berechtigungen zu erteilen, führen Sie die Schritte in diesem Abschnitt aus. Andernfalls können Sie mit [Schritt 3: Erstellen Sie eine Abonnementfilterrichtlinie auf Kontoebene](#) fortfahren.

Mit den Schritten in diesem Abschnitt wird eine IAM-Rolle erstellt, die davon ausgeht und überprüfen CloudWatch kann, ob das Absenderkonto berechtigt ist, einen Abonnementfilter für das Empfängerziel zu erstellen.

Führen Sie die Schritte in diesem Abschnitt im Sender-Konto aus. Die Rolle muss im Sender-Konto vorhanden sein, und Sie geben den ARN dieser Rolle im Abonnementfilter an. In diesem Beispiel ist das Sender-Konto 111111111111.

Um die IAM-Rolle zu erstellen, die für kontoübergreifende Protokollabonnements erforderlich ist, verwenden Sie AWS Organizations

1. Erstellen Sie die folgende Vertrauensrichtlinie in einer Datei namens `TrustPolicyForCWLSubscriptionFilter.json`. Erstellen Sie in einem Text-Editor die Richtliniendatei, verwenden Sie nicht die IAM-Konsole.

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

2. Erstellen Sie als Nächstes die IAM-Rolle, die diese Richtlinie verwendet. Notieren Sie sich den `Arn`-Wert, der vom Befehl zurückgegeben wird, Sie benötigen ihn später in diesem Verfahren. In

diesem Beispiel verwenden wir `CWLtoSubscriptionFilterRole` für den Namen der Rolle, die wir erstellen.

```
aws iam create-role \  
    --role-name CWLtoSubscriptionFilterRole \  
    --assume-role-policy-document file://~/TrustPolicyForCWLSubscriptionFilter.json
```

3. Erstellen Sie eine Berechtigungsrichtlinie, um die Aktionen zu definieren, die CloudWatch Logs auf Ihrem Konto ausführen kann.
 - a. Verwenden Sie zunächst einen Text-Editor, um die folgende Berechtigungsrichtlinie in einer Datei mit dem Namen zu erstellen `~/PermissionsForCWLSubscriptionFilter.json`.

```
{  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "logs:PutLogEvents",  
      "Resource": "arn:aws:logs:region:111111111111:log-group:LogGroupOnWhichSubscriptionFilterIsCreated:*"  
    }  
  ]  
}
```

- b. Geben Sie den folgenden Befehl ein, um die soeben erstellte Berechtigungsrichtlinie mit der Rolle zu verknüpfen, die Sie in Schritt 2 erstellt haben.

```
aws iam put-role-policy  
    --role-name CWLtoSubscriptionFilterRole  
    --policy-name Permissions-Policy-For-CWL-Subscription-filter  
    --policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

Wenn Sie fertig sind, können Sie mit [Schritt 3: Erstellen Sie eine Abonnementfilterrichtlinie auf Kontoebene](#) fortfahren.

Schritt 3: Erstellen Sie eine Abonnementfilterrichtlinie auf Kontoebene

Wenn Sie ein Ziel erstellt haben, kann das Empfängerkonto der Protokolldaten den Ziel-ARN (`arn:aws:logs:us-east-1:999999999999:destination:testDestination`) für andere AWS -Konten freigeben, damit diese Protokollereignisse an dasselbe Ziel senden können. Die anderen sendenden

Kontobenutzer erstellen einen Abonnementfilter für die jeweiligen Protokollgruppen für dieses Ziel. Der Abonnementfilter startet sofort den Fluss der Echtzeit-Protokolldaten aus der gewählten Protokollgruppe an das angegebene Ziel.

Note

Wenn Sie einer gesamten Organisation Berechtigungen für den Abonnementfilter gewähren, müssen Sie den ARN der IAM-Rolle verwenden, die Sie in [Schritt 2: \(Nur bei Verwendung einer Organisation\) Erstellen Sie eine IAM-Rolle](#) erstellt haben.

Im folgenden Beispiel wird eine Abonnementfilterrichtlinie auf Kontoebene in einem Absenderkonto erstellt. Der Filter ist dem Absenderkonto zugeordnet, 111111111111 sodass jedes Protokollereignis, das den Filter- und Auswahlkriterien entspricht, an das zuvor erstellte Ziel gesendet wird. Dieses Ziel kapselt einen Stream namens "". RecipientStream

Das `selection-criteria` Feld ist optional, aber wichtig, um Protokollgruppen, die zu einer unendlichen Protokollrekursion führen können, aus einem Abonnementfilter auszuschließen. Weitere Informationen zu diesem Problem und zur Bestimmung, welche Protokollgruppen ausgeschlossen werden sollen, finden Sie unter [Verhinderung der Rekursion von Protokollen](#). Derzeit ist NOT IN der einzige unterstützte Operator für `selection-criteria`.

```
aws logs put-account-policy \
  --policy-name "CrossAccountStreamsExamplePolicy" \
  --policy-type "SUBSCRIPTION_FILTER_POLICY" \
  --policy-document
'{"DestinationArn":"arn:aws:logs:region:999999999999:destination:testDestination",
"FilterPattern": "", "Distribution": "Random"}' \
  --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
"LogGroupToExclude2"]' \
  --scope "ALL"
```

Die Protokollgruppen des Absenderkontos und das Ziel müssen sich in derselben AWS Region befinden. Das Ziel kann jedoch auf eine AWS Ressource verweisen, z. B. auf einen Amazon Kinesis Data Streams Streams-Stream, der sich in einer anderen Region befindet.

Validierung des Ablaufs von Protokollereignissen

Nachdem Sie die Abonnementfilterrichtlinie auf Kontoebene erstellt haben, leitet CloudWatch Logs alle eingehenden Protokollereignisse, die dem Filtermuster und den Auswahlkriterien entsprechen, an

den Stream weiter, der im Zielstream mit dem Namen "" gekapselt ist. RecipientStream Der Besitzer des Ziels kann überprüfen, ob dies der Fall ist, indem er den get-shard-iterator Befehl aws kinesis verwendet, um einen Amazon Kinesis Data Streams Streams-Shard abzurufen, und den Befehl aws kinesis get-records verwendet, um einige Amazon Kinesis Data Streams Streams-Datensätze abzurufen:

```
aws kinesis get-shard-iterator \
  --stream-name RecipientStream \
  --shard-id shardId-000000000000 \
  --shard-iterator-type TRIM_HORIZON

{
  "ShardIterator":
  "AAAAAAAAAAFGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+0IK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
}

aws kinesis get-records \
  --limit 10 \
  --shard-iterator
  "AAAAAAAAAAFGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+0IK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
```

Note

Möglicherweise müssen Sie den get-records Befehl einige Male erneut ausführen, bevor Amazon Kinesis Data Streams Daten zurückgibt.

Sie sollten eine Antwort mit einer Reihe von Amazon Kinesis Data Streams Streams-Datensätzen sehen. Das Datenattribut im Amazon Kinesis Data Streams Streams-Datensatz wird im GZIP-Format komprimiert und anschließend base64-codiert. Sie können die Rohdaten über die Befehlszeile mit dem folgenden Unix-Befehl überprüfen:

```
echo -n "<Content of Data>" | base64 -d | zcat
```

Die dekodierten und dekomprimierten base64-Daten sind als JSON mit folgender Struktur formatiert:

```
{
  "owner": "111111111111",
  "logGroup": "CloudTrail/logs",
  "logStream": "111111111111_CloudTrail/logs_us-east-1",
  "subscriptionFilters": [
    "RecipientStream"
  ],
  "messageType": "DATA_MESSAGE",
  "logEvents": [
    {
      "id": "3195310660696698337880902507980421114328961542429EXAMPLE",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
    }\",
    {
      "id": "3195310660696698337880902507980421114328961542429EXAMPLE",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
    }\",
    {
      "id": "3195310660696698337880902507980421114328961542429EXAMPLE",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
    }
  ]
}
```

Die wichtigsten Elemente der Datenstruktur sind die folgenden:

messageType

Datennachrichten verwenden den Typ "DATA_MESSAGE". Manchmal geben CloudWatch Logs Amazon Kinesis Data Streams Streams-Datensätze mit dem Typ „CONTROL_MESSAGE“ aus, hauptsächlich um zu überprüfen, ob das Ziel erreichbar ist.

owner

Die AWS Konto-ID der ursprünglichen Protokolldaten.

logGroup

Der Name der Protokollgruppe der ursprünglichen Protokolldaten.

logStream

Der Name des Protokoll-Stream der ursprünglichen Protokolldaten.

subscriptionFilters

Die Namenliste der Abonnementfilter, die mit den ursprünglichen Protokolldaten übereingestimmt haben.

logEvents

Die tatsächlichen Protokolldaten, die als Array von Protokollereignis-Datensätzen dargestellt werden. Die "id"-Eigenschaft ist eine eindeutige Kennung für jedes Protokollereignis.

Richtlinienebene

Die Ebene, auf der die Richtlinie durchgesetzt wurde. „ACCOUNT_LEVEL_POLICY“ ist die Filterrichtlinie `policyLevel` für Abonnements auf Kontoebene.

Ändern der Mitgliedschaft im Ziel zur Laufzeit

In manchen Situationen müssen Sie die Mitgliedschaft einiger Benutzer in einem Ziel, das Ihr Eigentum ist, hinzufügen oder entfernen. Sie können den Befehl `put-destination-policy` für Ihr Ziel mit einer neuen Zugriffsrichtlinie verwenden. Im folgenden Beispiel wird festgelegt, dass das zuvor hinzugefügte Konto 111111111111 keine Protokolldaten mehr sendet und das Konto 222222222222 aktiviert wird.

1. Rufen Sie die Richtlinie ab, die derzeit mit dem Ziel `TestDestination` verknüpft ist, und notieren Sie sich Folgendes: `AccessPolicy`

```
aws logs describe-destinations \
  --destination-name-prefix "testDestination"

{
  "Destinations": [
    {
      "DestinationName": "testDestination",
      "RoleArn": "arn:aws:iam::999999999999:role/CWLtoKinesisRole",
      "DestinationArn":
        "arn:aws:logs:region:999999999999:destination:testDestination",
```

```

    "TargetArn": "arn:aws:kinesis:region:999999999999:stream/RecipientStream",
    "AccessPolicy": "{\\"Version\\": \\"2012-10-17\\", \\"Statement\\":
[{\\"Sid\\": \\"\\", \\"Effect\\": \\"Allow\\", \\"Principal\\": {\\"AWS\\":
\\"111111111111\\"}, \\"Action\\": \\"logs:PutSubscriptionFilter\\", \\"Resource\\":
\\"arn:aws:logs:region:999999999999:destination:testDestination\\"}] }"
  }
]
}

```

2. Aktualisieren Sie die Richtlinie, sodass angezeigt wird, dass das Konto 111111111111 angehalten wurde und das Konto 222222222222 aktiviert wurde. Fügen Sie diese Richtlinie in die Datei ~/NewAccessPolicy.json ein:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "222222222222"
      },
      "Action": [
        "logs:PutSubscriptionFilter",
        "logs:PutAccountPolicy"
      ],
      "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination"
    }
  ]
}

```

3. Rufen Sie PutDestinationPolicy auf, um die in der NewAccessPolicy.json-Datei definierte Richtlinie dem Ziel zuzuordnen:

```

aws logs put-destination-policy \
--destination-name "testDestination" \
--access-policy file://~/NewAccessPolicy.json

```

Damit werden schließlich die Protokollereignisse von der Konto-ID 111111111111 deaktiviert. Protokollereignisse der Konto-ID 222222222222 werden an das Ziel übertragen, sobald der Eigentümer des Kontos 222222222222 einen Abonnementfilter erstellt.

Aktualisieren eines bestehenden kontoübergreifenden Abonnements

Wenn Sie derzeit über ein kontoübergreifendes Protokoll-Abonnement verfügen, bei dem das Zielkonto Berechtigungen nur bestimmten Senderkonten gewährt, und Sie dieses Abonnement aktualisieren möchten, damit das Zielkonto Zugriff auf alle Konten in einer Organisation gewährt, führen Sie die Schritte in diesem Abschnitt aus.

Themen

- [Schritt 1: Aktualisieren der Abonnementfilter](#)
- [Schritt 2: Aktualisieren der bestehenden Richtlinie für die Zugriffsrichtlinie](#)

Schritt 1: Aktualisieren der Abonnementfilter

Note

Dieser Schritt ist nur für kontoübergreifende Abonnements für Protokolle erforderlich, die von den in [Protokollierung aktivieren von AWS service](#) aufgeführten Services erstellt werden. Wenn Sie nicht mit Protokollen arbeiten, die von einer dieser Protokollgruppen erstellt wurden, können Sie zu [Schritt 2: Aktualisieren der bestehenden Richtlinie für die Zugriffsrichtlinie](#) wechseln.

In bestimmten Fällen müssen Sie die Abonnementfilter in allen Absenderkonten aktualisieren, die Protokolle an das Zielkonto senden. Das Update fügt eine IAM-Rolle hinzu, die davon ausgehen und überprüfen CloudWatch kann, dass das Absenderkonto berechtigt ist, Protokolle an das Empfängerkonto zu senden.

Befolgen Sie die Schritte in diesem Abschnitt für jedes Senderkonto, das Sie aktualisieren möchten, um die Organisations-ID für die kontoübergreifenden Abonnementberechtigungen zu verwenden.

In den Beispielen in diesem Abschnitt wurden für zwei Konten, 111111111111 und 222222222222, bereits Abonnementfilter erstellt, um Protokolle an Konto 999999999999 zu senden. Die vorhandenen Abonnementfilterwerte lauten wie folgt:

```
## Existing Subscription Filter parameter values
{
  "DestinationArn": "arn:aws:logs:region:999999999999:destination:testDestination",
  "FilterPattern": "{$.userIdentity.type = Root}",
  "Distribution": "Random"
}
```

Wenn Sie die aktuellen Parameterwerte des Abonnementfilters suchen müssen, geben Sie den folgenden Befehl ein.

```
aws logs describe-account-policies \
--policy-type "SUBSCRIPTION_FILTER_POLICY" \
--policy-name "CrossAccountStreamsExamplePolicy"
```

Um einen Abonnementfilter zu aktualisieren, um ab sofort Organisation IDs für kontoübergreifende Protokollberechtigungen zu verwenden

1. Erstellen Sie die folgende Vertrauensrichtlinie in einer Datei namens `~/TrustPolicyForCWL.json`. Erstellen Sie in einem Text-Editor die Richtliniendatei, verwenden Sie nicht die IAM-Konsole.

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

2. Erstellen Sie als Nächstes die IAM-Rolle, die diese Richtlinie verwendet. Notieren Sie sich den Arn-Wert des Arn-Werts, der vom Befehl zurückgegeben wird, den Sie später in diesem Verfahren benötigen. In diesem Beispiel verwenden wir `CWLtoSubscriptionFilterRole` für den Namen der Rolle, die wir erstellen.

```
aws iam create-role
\ --role-name CWLtoSubscriptionFilterRole
\ --assume-role-policy-document file://~/TrustPolicyForCWL.json
```

3. Erstellen Sie eine Berechtigungsrichtlinie, um die Aktionen zu definieren, die CloudWatch Logs auf Ihrem Konto ausführen kann.

- a. Verwenden Sie zunächst einen Text-Editor, um die folgende Berechtigungsrichtlinie in einer Datei mit dem Namen zu erstellen `/PermissionsForCWLSubscriptionFilter.json`.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:PutLogEvents",
      "Resource": "arn:aws:logs:region:111111111111:log-
group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
    }
  ]
}
```

- b. Geben Sie den folgenden Befehl ein, um die soeben erstellte Berechtigungsrichtlinie mit der Rolle zu verknüpfen, die Sie in Schritt 2 erstellt haben.

```
aws iam put-role-policy
  --role-name CWLtoSubscriptionFilterRole
  --policy-name Permissions-Policy-For-CWL-Subscription-filter
  --policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

4. Geben Sie den folgenden Befehl ein, um die Abonnementfilterrichtlinie zu aktualisieren.

```
aws logs put-account-policy \
  --policy-name "CrossAccountStreamsExamplePolicy" \
  --policy-type "SUBSCRIPTION_FILTER_POLICY" \
  --policy-document
'{"DestinationArn":"arn:aws:logs:region:999999999999:destination:testDestination",
"FilterPattern": "{$.userIdentity.type = Root}", "Distribution": "Random"}' \
  --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
"LogGroupToExclude2"]' \
  --scope "ALL"
```

Schritt 2: Aktualisieren der bestehenden Richtlinie für die Zugriffsrichtlinie

Nachdem Sie die Abonnementfilter in allen Senderkonten aktualisiert haben, können Sie die Zielzugriffsrichtlinie im Empfängerkonto aktualisieren.

In den folgenden Beispielen ist das Empfängerkonto 999999999999 und das Ziel hat den Namen `testDestination`.

Das Update ermöglicht es allen Konten, die Teil der Organisation mit der ID `o-1234567890` sind, Protokolle an das Empfängerkonto zu senden. Nur die Konten, für die Abonnementfilter erstellt wurden, senden tatsächlich Protokolle an das Empfängerkonto.

So aktualisieren Sie die Zielzugriffsrichtlinie im Empfängerkonto, um eine Organisations-ID für Berechtigungen zu verwenden

1. Verwenden Sie im Empfängerkonto einen Texteditor, um eine `~/AccessPolicy.json`-Datei mit dem folgenden Inhalt zu erstellen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "logs:PutSubscriptionFilter",
        "logs:PutAccountPolicy"
      ],
      "Resource": "arn:aws:logs:us-east-1:999999999999:destination:testDestination",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": [
            "o-1234567890"
          ]
        }
      }
    }
  ]
}
```

2. Geben Sie den folgenden Befehl ein, um die Richtlinie, die Sie soeben erstellt haben, dem vorhandenen Ziel anzufügen. Um ein Ziel so zu aktualisieren, dass es eine Zugriffsrichtlinie mit

einer Organisations-ID anstelle einer Zugriffsrichtlinie verwendet, in der ein bestimmtes AWS Konto aufgeführt ist IDs, geben Sie den `force` Parameter an.

 Warning

Wenn Sie mit Protokollen arbeiten, die von einem AWS Dienst gesendet wurden, der unter aufgeführt ist [Protokollierung aktivieren von AWS service](#), müssen Sie, bevor Sie diesen Schritt ausführen, zuerst die Abonnementfilter in allen Absenderkonten aktualisiert haben, wie unter beschrieben [Schritt 1: Aktualisieren der Abonnementfilter](#).

```
aws logs put-destination-policy
\ --destination-name "testDestination"
\ --access-policy file://~/AccessPolicy.json
\ --force
```

Kontoübergreifende, regionsübergreifende Abonnements auf Kontoebene mit Firehose

Wenn Sie Protokolldaten für mehrere Konten freigeben möchten, müssen Sie Sender und Receiver der Protokolldaten festlegen:

- Absender der Protokolldaten — Ruft die Zielinformationen vom Empfänger ab und teilt Logs mit, dass CloudWatch Logs bereit ist, seine Protokollereignisse an das angegebene Ziel zu senden. In den Verfahren im Rest dieses Abschnitts wird der Absender der Protokolldaten mit der fiktiven AWS Kontonummer 1111111111 angezeigt.
- Empfänger von Protokolldaten — richtet ein Ziel ein, das einen Amazon Kinesis Data Streams Streams-Stream kapselt und CloudWatch Logs darüber informiert, dass der Empfänger Protokolldaten empfangen möchte. Der Empfänger gibt dann die Informationen über sein Ziel für den Absender frei. In den Verfahren im Rest dieses Abschnitts wird der Empfänger der Protokolldaten mit der fiktiven AWS Kontonummer 222222222222 angezeigt.

Das Beispiel in diesem Abschnitt verwendet einen Firehose-Lieferstream mit Amazon S3 S3-Speicher. Sie können Firehose-Lieferstreams auch mit unterschiedlichen Einstellungen einrichten. Weitere Informationen finden Sie unter [Firehose Delivery Stream erstellen](#).

 Note

Die Protokollgruppe und das Ziel müssen sich in derselben AWS Region befinden. Die AWS Ressource, auf die das Ziel verweist, kann sich jedoch in einer anderen Region befinden.

 Note

Der Firehose-Abonnementfilter für dasselbe Konto und denselben regionsübergreifenden Lieferstream wird unterstützt.

Themen

- [Schritt 1: Erstellen Sie einen Firehose-Lieferstream](#)
- [Schritt 2: Erstellen eines Ziels](#)
- [Schritt 3: Erstellen Sie eine Abonnementfilterrichtlinie auf Kontoebene](#)
- [Validierung des Flusses von Protokollereignissen](#)
- [Ändern der Mitgliedschaft im Ziel zur Laufzeit](#)

Schritt 1: Erstellen Sie einen Firehose-Lieferstream

 Important

Bevor Sie die folgenden Schritte ausführen, müssen Sie eine Zugriffsrichtlinie verwenden, damit Firehose auf Ihren Amazon S3 S3-Bucket zugreifen kann. Weitere Informationen finden Sie unter [Zugriffskontrolle](#) im Amazon Data Firehose Developer Guide.

Alle Schritte in diesem Abschnitt (Schritt 1) müssen im Konto des Protokolldatenempfängers ausgeführt werden.

USA Ost (Nord-Virginia) wird in den folgenden Beispielbefehlen verwendet. Ersetzen Sie diese Region durch die richtige Region für Ihre Bereitstellung.

Um einen Firehose-Lieferstream zu erstellen, der als Ziel verwendet werden soll

1. Erstellen eines Amazon-S3-Buckets:

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --create-bucket-configuration
LocationConstraint=us-east-1
```

2. Erstellen Sie die IAM-Rolle, die Firehose die Berechtigung erteilt, Daten in den Bucket zu legen.

- a. Verwenden Sie zunächst einen Text-Editor zum Erstellen einer Vertrauensrichtlinie in einer Datei `~/TrustPolicyForFirehose.json`.

```
{ "Statement": { "Effect": "Allow", "Principal": { "Service":
"firehose.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition":
{ "StringEquals": { "sts:ExternalId": "222222222222" } } } }
```

- b. Erstellen Sie die IAM-Rolle und geben Sie die Datei mit der Vertrauensrichtlinie an, die Sie gerade erstellt haben.

```
aws iam create-role \
  --role-name FirehosetoS3Role \
  --assume-role-policy-document file://~/TrustPolicyForFirehose.json
```

- c. Die Ausgabe für den Befehl sieht dann wie folgt aus. Notieren Sie sich den Rollennamen und die Rollen-ARN.

```
{
  "Role": {
    "Path": "/",
    "RoleName": "FirehosetoS3Role",
    "RoleId": "AR0AR3BXASEKW7K635M53",
    "Arn": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
    "CreateDate": "2021-02-02T07:53:10+00:00",
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Effect": "Allow",
        "Principal": {
          "Service": "firehose.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
          "StringEquals": {
            "sts:ExternalId": "222222222222"
          }
        }
      }
    }
  }
}
```

```

    }
  }
}

```

3. Erstellen Sie eine Berechtigungsrichtlinie, um die Aktionen zu definieren, die Firehose in Ihrem Konto ausführen kann.
 - a. Verwenden Sie zunächst einen Text-Editor, um die folgende Berechtigungsrichtlinie in einer Datei mit dem Namen zu erstellen `~/PermissionsForFirehose.json`. Abhängig von Ihrem Anwendungsfall müssen Sie dieser Datei möglicherweise weitere Berechtigungen hinzufügen.

```

{
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:ListBucket"
    ],
    "Resource": [
      "arn:aws:s3:::amzn-s3-demo-bucket",
      "arn:aws:s3:::amzn-s3-demo-bucket/*"
    ]
  }]
}

```

- b. Geben Sie den folgenden Befehl ein, um die soeben erstellte Berechtigungsrichtlinie der IAM-Rolle zuzuordnen.

```

aws iam put-role-policy --role-name FirehoseToS3Role --policy-name
Permissions-Policy-For-Firehose-To-S3 --policy-document file://~/
PermissionsForFirehose.json

```

4. Geben Sie den folgenden Befehl ein, um den Firehose-Lieferstream zu erstellen. Ersetzen Sie *my-role-arn* und *amzn-s3-demo-bucket2-arn* durch die richtigen Werte für Ihre Bereitstellung.

```

aws firehose create-delivery-stream \
  --delivery-stream-name 'my-delivery-stream' \
  --s3-destination-configuration \

```

```
'{"RoleARN": "arn:aws:iam::222222222222:role/FirehoseToS3Role", "BucketARN":  
"arn:aws:s3:::amzn-s3-demo-bucket"}'
```

Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```
{  
  "DeliveryStreamARN": "arn:aws:firehose:us-east-1:222222222222:deliverystream/  
my-delivery-stream"  
}
```

Schritt 2: Erstellen eines Ziels

Important

Alle Schritte in diesem Verfahren müssen im Konto des Protokolldatenempfängers ausgeführt werden.

Wenn das Ziel erstellt wurde, sendet CloudWatch Logs im Namen des Empfängerkontos eine Testnachricht an das Ziel. Wenn der Abonnementfilter später aktiv ist, sendet CloudWatch Logs im Namen des Quellkontos Protokollereignisse an das Ziel.

So erstellen Sie ein Ziel

1. Warten Sie, bis der Firehose-Stream, in dem Sie ihn erstellt haben, aktiv [Schritt 1: Erstellen Sie einen Firehose-Lieferstream](#) wird. Sie können den folgenden Befehl verwenden, um das StreamDescription zu überprüfen. StreamStatusEigentum.

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
```

Beachten Sie außerdem die DeliveryStreamDescription. DeliveryStreamARN-Wert, da Sie ihn in einem späteren Schritt verwenden müssen. Beispielausgabe dieses Befehls:

```
{  
  "DeliveryStreamDescription": {  
    "DeliveryStreamName": "my-delivery-stream",  
    "DeliveryStreamARN": "arn:aws:firehose:us-  
east-1:222222222222:deliverystream/my-delivery-stream",  
    "DeliveryStreamStatus": "ACTIVE",
```

```

    "DeliveryStreamEncryptionConfiguration": {
      "Status": "DISABLED"
    },
    "DeliveryStreamType": "DirectPut",
    "VersionId": "1",
    "CreateTimestamp": "2021-02-01T23:59:15.567000-08:00",
    "Destinations": [
      {
        "DestinationId": "destinationId-000000000001",
        "S3DestinationDescription": {
          "RoleARN": "arn:aws:iam::222222222222:role/FirehoseToS3Role",
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
          "BufferingHints": {
            "SizeInMBs": 5,
            "IntervalInSeconds": 300
          },
          "CompressionFormat": "UNCOMPRESSED",
          "EncryptionConfiguration": {
            "NoEncryptionConfig": "NoEncryption"
          },
          "CloudWatchLoggingOptions": {
            "Enabled": false
          }
        },
        "ExtendedS3DestinationDescription": {
          "RoleARN": "arn:aws:iam::222222222222:role/FirehoseToS3Role",
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
          "BufferingHints": {
            "SizeInMBs": 5,
            "IntervalInSeconds": 300
          },
          "CompressionFormat": "UNCOMPRESSED",
          "EncryptionConfiguration": {
            "NoEncryptionConfig": "NoEncryption"
          },
          "CloudWatchLoggingOptions": {
            "Enabled": false
          },
          "S3BackupMode": "Disabled"
        }
      }
    ],
    "HasMoreDestinations": false
  }

```

```
}
```

Es kann einige Minuten dauern, bis der Bereitstellungsdatenstrom im aktiven Status angezeigt wird.

2. Wenn der Lieferstream aktiv ist, erstellen Sie die IAM-Rolle, die CloudWatch Logs die Erlaubnis erteilt, Daten in Ihren Firehose-Stream einzufügen. Zunächst müssen Sie eine Vertrauensrichtlinie in einer Datei `TrustPolicyFor~/.CWL.json` erstellen. Verwenden Sie einen Text-Editor, um diese Richtlinie zu erstellen. Weitere Informationen zu CloudWatch Logs-Endpunkten finden Sie unter [Amazon CloudWatch Logs-Endpunkte und Kontingente](#).

Diese Richtlinie enthält einen globalen Bedingungskontextschlüssel `aws:SourceArn`, der das `sourceAccountId` angibt, um das Confused-Deputy-Problem zu vermeiden. Wenn Sie die Quell-Kontonummer beim ersten Aufruf noch nicht kennen, empfehlen wir Ihnen, die Ziel-ARN in das Quell-ARN-Feld einzutragen. Bei den folgenden Aufrufen sollten Sie als Quell-ARN den tatsächlichen Quell-ARN angeben, den Sie beim ersten Aufruf ermittelt haben. Weitere Informationen finden Sie unter [Confused-Deputy-Prävention](#).

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "Service": "logs.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringLike": {
        "aws:SourceArn": [
          "arn:aws:logs:region:sourceAccountId:*",
          "arn:aws:logs:region:recipientAccountId:*"
        ]
      }
    }
  }
}
```

3. Verwenden Sie den Befehl `aws iam create-role`, um die IAM-Rolle zu erstellen und geben Sie die Datei mit der Vertrauensrichtlinie an, die Sie gerade erstellt haben.

```
aws iam create-role \
  --role-name CWLtoKinesisFirehoseRole \
```



```
--assume-role-policy-document file://~/TrustPolicyForCWL.json
```

Dies ist eine Beispielausgabe. Notieren Sie den zurückgegebenen `Role.Arn`-Wert, da Sie ihn in einem späteren Schritt benötigen.

```
{
  "Role": {
    "Path": "/",
    "RoleName": "CWLtoKinesisFirehoseRole",
    "RoleId": "AR0AR3BXASEKYJYWF243H",
    "Arn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
    "CreateDate": "2023-02-02T08:10:43+00:00",
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Effect": "Allow",
        "Principal": {
          "Service": "logs.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
          "StringLike": {
            "aws:SourceArn": [
              "arn:aws:logs:region:sourceAccountId:*",
              "arn:aws:logs:region:recipientAccountId:*"
            ]
          }
        }
      }
    }
  }
}
```

- Erstellen Sie eine Berechtigungsrichtlinie, um zu definieren, welche Aktionen CloudWatch Logs auf Ihrem Konto ausführen kann. Verwenden Sie zunächst einen Texteditor, um eine Berechtigungsrichtlinie in einer Datei `~/PermissionsForCWL.json` zu erstellen:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["firehose:*"],
      "Resource": ["arn:aws:firehose:region:222222222222:*"]
    }
  ]
}
```

```
    }
  ]
}
```

5. Verknüpfen Sie mit dem folgenden Befehl die Berechtigungsrichtlinie mit der Rolle:

```
aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-name
Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json
```

6. Nachdem sich der Firehose-Lieferstream im aktiven Zustand befindet und Sie die IAM-Rolle erstellt haben, können Sie das CloudWatch Logs-Ziel erstellen.
- a. In diesem Schritt wird keine Zugriffsrichtlinie mit Ihrem Ziel verknüpft. Es ist zudem erst der erste von zwei Schritten zum Erstellen eines Ziels. Notieren Sie sich die in der Nutzlast zurückgegebene ARN des neuen Zielortes, da Sie diese in einem späteren Schritt als `destination.arn` verwenden werden.

```
aws logs put-destination \

    --destination-name "testFirehoseDestination" \
    --target-arn "arn:aws:firehose:us-east-1:222222222222:deliverystream/my-
delivery-stream" \
    --role-arn "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole"

{
  "destination": {
    "destinationName": "testFirehoseDestination",
    "targetArn": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream",
    "roleArn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
    "arn": "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"}
}
```

- b. Verknüpfen Sie, nachdem der vorherige Schritt abgeschlossen ist, im Empfängerkonto der Protokolldaten (222222222222) eine Zugriffsrichtlinie mit dem Ziel. Diese Richtlinie ermöglicht dem Sender-Konto der Protokolldaten (111111111111), auf das Ziel nur im Empfängerkonto der Protokolldaten (222222222222) zuzugreifen. Sie können einen Texteditor verwenden, um diese Richtlinie in die `~/AccessPolicy.json` Datei einzufügen:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": ["logs:PutSubscriptionFilter", "logs:PutAccountPolicy"],
      "Resource": "arn:aws:logs:us-east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}
```

- c. Damit wird eine Richtlinie erstellt, die bestimmt, wer Schreibzugriff auf das Ziel hat. In dieser Richtlinie müssen die `logs:PutAccountPolicy` Aktionen `logs:PutSubscriptionFilter` und für den Zugriff auf das Ziel angegeben werden. Kontoübergreifende Benutzer verwenden die `PutAccountPolicy` Aktionen `PutSubscriptionFilter` und, um Protokollereignisse an das Ziel zu senden.

```
aws logs put-destination-policy \
  --destination-name "testFirehoseDestination" \
  --access-policy file://~/AccessPolicy.json
```

Schritt 3: Erstellen Sie eine Abonnementfilterrichtlinie auf Kontoebene

Wechseln Sie zum Senderkonto, das in diesem Beispiel 111111111111 lautet. Sie werden nun die Abonnementfilterrichtlinie auf Kontoebene im sendenden Konto erstellen. In diesem Beispiel bewirkt der Filter, dass jedes Protokollereignis, das die Zeichenfolge enthält, `ERROR` in allen außer zwei Protokollgruppen an das Ziel gesendet wird, das Sie zuvor erstellt haben.

```
aws logs put-account-policy \
  --policy-name "CrossAccountFirehoseExamplePolicy" \
  --policy-type "SUBSCRIPTION_FILTER_POLICY" \
```

```
--policy-document '{"DestinationArn":"arn:aws:logs:us-east-1:222222222222:destination:testFirehoseDestination", "FilterPattern":
"{$.userIdentity.type = AssumedRole}", "Distribution": "Random"}' \
--selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
"LogGroupToExclude2"]' \
--scope "ALL"
```

Die Protokollgruppen des sendenden Kontos und das Ziel müssen sich in derselben AWS Region befinden. Das Ziel kann jedoch auf eine AWS Ressource verweisen, z. B. auf einen Firehose-Stream, der sich in einer anderen Region befindet.

Validierung des Flusses von Protokollereignissen

Nachdem Sie den Abonnementfilter erstellt haben, leitet CloudWatch Logs alle eingehenden Protokollereignisse, die dem Filtermuster und den Auswahlkriterien entsprechen, an den Firehose-Lieferstream weiter. Die Daten werden in Ihrem Amazon S3 S3-Bucket basierend auf dem Zeitpufferintervall angezeigt, das für den Firehose-Lieferstream festgelegt ist. Sobald genügend Zeit abgelaufen ist, können Sie die Daten im Amazon-S3-Bucket überprüfen. Geben Sie zur Prüfung des Buckets den folgenden Befehl ein:

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket'
```

Die Ausgabe des Befehls sieht dann wie folgt aus:

```
{
  "Contents": [
    {
      "Key": "2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba",
      "LastModified": "2023-02-02T09:00:26+00:00",
      "ETag": "\"EXAMPLEa817fb88fc770b81c8f990d\"",
      "Size": 198,
      "StorageClass": "STANDARD",
      "Owner": {
        "DisplayName": "firehose+2test",
        "ID": "EXAMPLE27fd05889c665d2636218451970ef79400e3d2aecca3adb1930042e0"
      }
    }
  ]
}
```

Sie können dann ein bestimmtes Objekt aus dem Bucket abrufen, indem Sie den folgenden Befehl eingeben. Ersetzen Sie den Wert von `key` mit dem Wert, den Sie im vorherigen Befehl gefunden haben.

```
aws s3api get-object --bucket 'amzn-s3-demo-bucket' --key '2021/02/02/08/my-delivery-stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba' testfile.gz
```

Die Daten im Amazon-S3-Objekt sind im GZIP-Format komprimiert. Sie können die Rohdaten über die Befehlszeile mit einem der folgenden Befehlen überprüfen:

Linux:

```
zcat testfile.gz
```

macOS:

```
zcat <testfile.gz
```

Ändern der Mitgliedschaft im Ziel zur Laufzeit

In manchen Situationen müssen Sie Protokollsender in einem Ziel, das Ihr Eigentum ist, hinzufügen oder entfernen. Mit der neuen Zugriffsrichtlinie können Sie die `PutAccountPolicy` Aktionen `PutDestinationPolicy` und auf Ihrem Ziel verwenden. Im folgenden Beispiel wird festgelegt, dass das zuvor hinzugefügte Konto 111111111111 keine Protokolldaten mehr senden kann und das Konto 333333333333 aktiviert wird.

1. Rufen Sie die Richtlinie ab, die derzeit mit dem Ziel `TestDestination` verknüpft ist, und notieren Sie sich Folgendes: `AccessPolicy`

```
aws logs describe-destinations \
  --destination-name-prefix "testFirehoseDestination"
```

Die zurückgegebenen Daten könnten so aussehen.

```
{
  "destinations": [
    {
      "destinationName": "testFirehoseDestination",
      "targetArn": "arn:aws:firehose:us-east-1:222222222222:deliverystream/my-delivery-stream",

```

```

        "roleArn": "arn:aws:iam:: 222222222222:role/CWLtoKinesisFirehoseRole",
        "accessPolicy": "{\n  \"Version\" : \"2012-10-17\",\n  \"Statement\n\" : [\n    {\n      \"Sid\" : \"\",\n      \"Effect\" : \"Allow\",\n      \"Principal\" : {\n        \"AWS\" : \"111111111111 \"\n      },\n      \"Action\n\" : \"logs:PutSubscriptionFilter\",\n      \"Resource\" : \"arn:aws:logs:us-east-1:222222222222:destination:testFirehoseDestination\"\n    }\n  ]\n}\n",
        "arn": "arn:aws:logs:us-east-1:222222222222:destination:testFirehoseDestination",
        "creationTime": 1612256124430
      }
    ]
  }

```

2. Aktualisieren Sie die Richtlinie dementsprechend, dass das Konto 111111111111 angehalten wurde und das Konto 333333333333 aktiviert wurde. Fügen Sie diese Richtlinie in die Datei `~/NewAccessPolicy.json` ein:

JSON

```

{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "333333333333 "
      },
      "Action" : ["logs:PutSubscriptionFilter", "logs:PutAccountPolicy"],
      "Resource" : "arn:aws:logs:us-east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}

```

3. Verwenden Sie den folgenden Befehl, um die in der `NewAccessPolicyJSON`-Datei definierte Richtlinie dem Ziel zuzuordnen:

```

aws logs put-destination-policy \
  --destination-name "testFirehoseDestination" \

```

```
--access-policy file://~/NewAccessPolicy.json
```

Damit werden schließlich die Protokollereignisse von der Konto-ID 111111111111 deaktiviert. Protokollereignisse der Konto-ID 333333333333 werden an das Ziel übertragen, sobald der Eigentümer des Kontos 333333333333 einen Abonnementfilter erstellt.

Confused-Deputy-Prävention

Das Problem des verwirrten Stellvertreters ist ein Sicherheitsproblem, bei dem eine Entität, die keine Berechtigung zur Durchführung einer Aktion hat, eine privilegiere Entität zur Durchführung der Aktion zwingen kann. In AWS kann ein dienstübergreifendes Identitätswechsel zu einem Problem mit dem verwirrten Stellvertreter führen. Ein dienstübergreifender Identitätswechsel kann auftreten, wenn ein Dienst (der Anruf-Dienst) einen anderen Dienst anruft (den aufgerufenen Dienst). Der Anruf-Dienst kann so manipuliert werden, dass er seine Berechtigungen verwendet, um auf die Ressourcen eines anderen Kunden zu reagieren, auf die er sonst nicht zugreifen dürfte. Um dies zu verhindern, AWS bietet Tools, mit denen Sie Ihre Daten für alle Dienste mit Dienstprinzipalen schützen können, denen Zugriff auf Ressourcen in Ihrem Konto gewährt wurde.

Wir empfehlen [aws:SourceArn](#), die Kontextschlüssel, [aws:SourceAccount](#), [aws:SourceOrgID](#), und die [aws:SourceOrgPaths](#) globalen Bedingungsschlüssel in Ressourcenrichtlinien zu verwenden, um die Berechtigungen einzuschränken, die der Ressource einen anderen Dienst gewähren. Verwenden Sie `aws:SourceArn`, um nur einer Ressource den serviceübergreifenden Zugriff zuzuordnen. Verwenden Sie `aws:SourceAccount`, um jede Ressource in diesem Konto der serviceübergreifenden Nutzung zuzuordnen. Verwenden Sie `aws:SourceOrgID`, damit alle Ressourcen von allen Konten innerhalb einer Organisation der serviceübergreifenden Nutzung zugeordnet werden können. Verwenden Sie `aws:SourceOrgPaths`, um beliebige Ressourcen von Konten innerhalb eines AWS Organizations -Pfades der serviceübergreifenden Nutzung zuzuordnen. Weitere Informationen zur Verwendung und zum Verständnis von Pfaden finden Sie unter [Grundlegendes zum AWS Organizations Entitätspfad](#).

Der effektivste Weg, um sich vor dem Confused-Deputy-Problem zu schützen, ist die Verwendung des globalen Bedingungskontext-Schlüssels `aws:SourceArn` mit dem vollständigen ARN der Ressource. Wenn Sie den vollständigen ARN der Ressource nicht kennen oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den globalen Kontextbedingungsschlüssel `aws:SourceArn` mit Platzhalterzeichen (*) für die unbekannten Teile des ARN. Beispiel, `arn:aws:service:*:123456789012:*`.

Wenn der `aws:SourceArn`-Wert die Konto-ID nicht enthält, z. B. einen Amazon-S3-Bucket-ARN, müssen Sie sowohl `aws:SourceAccount` als auch `aws:SourceArn` verwenden, um Berechtigungen einzuschränken.

Um sich vor dem Confused-Deputy-Problem im großen Maßstab zu schützen, verwenden Sie den globalen Bedingungskontextschlüssel `aws:SourceOrgID` oder `aws:SourceOrgPaths` mit der Organisations-ID oder dem Organisationspfad der Ressource in Ihren ressourcenbasierten Richtlinien. Richtlinien, die den Schlüssel `aws:SourceOrgID` oder `aws:SourceOrgPaths` enthalten, schließen automatisch die richtigen Konten ein und Sie müssen die Richtlinien nicht manuell aktualisieren, wenn Sie Konten in Ihrer Organisation hinzufügen, entfernen oder verschieben.

Die dokumentierten Richtlinien für die Gewährung des Zugriffs auf CloudWatch Logs zum Schreiben von Daten in Amazon Kinesis Data Streams und Firehose in [Schritt 1: Erstellen eines Ziels](#) und [Schritt 2: Erstellen eines Ziels](#) zeigen, wie Sie den `aws:SourceArn` globalen Bedingungskontextschlüssel verwenden können, um das Problem des verwirrten Stellvertreters zu verhindern.

Verhinderung der Rekursion von Protokollen

Bei Abonnementfiltern besteht das Risiko, dass eine unendliche Protokollrekursion entsteht, was zu einem starken Anstieg der Abrechnung für Datenerfassung sowohl in den CloudWatch Protokollen als auch in Ihrem Zielverzeichnis führen kann, wenn dies nicht sogar verhindert wird. Dies kann der Fall sein, wenn ein Abonnementfilter einer Protokollgruppe zugeordnet ist, die aufgrund Ihres Workflows für die Abonnementzustellung Protokollereignisse empfängt. Die in die Protokollgruppe aufgenommenen Protokolle werden an das Ziel übermittelt, wodurch die Protokollgruppe mehr Protokolle aufnimmt, die dann wieder an das Ziel weitergeleitet werden, wodurch eine Rekursionsschleife entsteht.

Stellen Sie sich zum Beispiel einen Abonnementfilter mit dem Ziel Firehose vor, der Protokollereignisse an Amazon S3 übermittelt. Darüber hinaus gibt es auch eine Lambda-Funktion, die neue Ereignisse verarbeitet, die an Amazon S3 übermittelt werden, und einige Protokolle selbst erstellt. Wenn der Abonnementfilter auf die Protokollgruppe der Lambda-Funktion angewendet wird, werden die von der Funktion erzeugten Protokollereignisse an Firehose und Amazon S3 am Ziel weitergeleitet, die dann die Funktion erneut aufrufen, wodurch mehr Protokolle erstellt und an Firehose und Amazon S3 weitergeleitet werden, was zu einem erneuten Aufruf der Funktion führt usw. Dies geschieht in einer Endlosschleife, was zu einer unerwarteten Erhöhung der Gebühren für Log-Ingestion, Firehose und Amazon S3 führt.

Wenn die Lambda-Funktion an eine VPC angehängt ist, deren Flow-Logs für CloudWatch Logs aktiviert sind, kann die Log-Gruppe der VPC ebenfalls eine Log-Rekursion auslösen.

Wir empfehlen, dass Sie keine Abonnementfilter auf Protokollgruppen anwenden, die Teil Ihres Workflows für die Abonnementzustellung sind. Verwenden Sie für Abonnementfilter auf Kontoebene den `selectionCriteria` Parameter in der `PutAccountPolicy` API, um diese Protokollgruppen von der Richtlinie auszuschließen.

Wenn Sie Protokollgruppen ausschließen, sollten Sie die folgenden AWS Dienste in Betracht ziehen, die Protokolle erstellen und Teil Ihrer Workflows für die Abonnementzustellung sein können:

- Amazon EC2 mit Fargate
- Lambda
- AWS Step Functions
- Amazon VPC-Flow-Logs, die für CloudWatch Logs aktiviert sind

 Note

Protokollereignisse, die von der Protokollgruppe eines Lambda-Ziels erzeugt wurden, werden für eine Abonnementfilterrichtlinie auf Kontoebene nicht an die Lambda-Funktion zurückgeleitet. In diesem Fall ist es für Kontoabonnementrichtlinien nicht erforderlich, die Verwendung `selectionCriteria` der Protokollgruppe der Ziel-Lambda-Funktion auszuschließen.

Filtermustersyntax für Metrikfilter, Abonnementfilter, Filterprotokollereignisse und Live Tail

Note

Informationen darüber, wie Sie Ihre Protokollgruppen mit der Abfragesprache von Amazon CloudWatch Logs Insights abfragen, finden Sie unter [CloudWatch Abfragesyntax in Logs Insights](#).

Mit CloudWatch Logs können Sie [Metrikfilter](#) verwenden, um Protokolldaten in aussagekräftige Metriken umzuwandeln, [Abonnementfilter](#), um Protokollereignisse an andere AWS Dienste weiterzuleiten, [Protokollereignisse filtern](#), um nach Protokollereignissen zu suchen, und [Live Tail](#), um Ihre Protokolle interaktiv in Echtzeit anzuzeigen, während sie aufgenommen werden.

Filtermuster bilden die Syntax, die Metrikfilter, Abonnementfilter, Protokollereignisse und Live Tail verwenden, um Begriffe in Protokollereignissen abzugleichen. Begriffe können Wörter, exakte Phrasen oder numerische Werte sein. Reguläre Ausdrücke (Regex) können verwendet werden, um eigenständige Filtermuster zu erstellen, oder sie können in JSON- und durch Leerzeichen getrennte Filtermuster integriert werden.

Erstellen Sie Filtermuster mit den Begriffen, die Sie abgleichen möchten. Filtermuster geben nur die Protokollereignisse zurück, die die von Ihnen definierten Begriffe enthalten. Sie können Filtermuster in der CloudWatch Konsole testen.

Themen

- [Unterstützte Syntax für reguläre Ausdrücke \(Regex\)](#)
- [Verwenden von Filtermustern zum Abgleichen von Begriffen mit einem regulären Ausdruck \(Regex\)](#)
- [Verwenden von Filtermustern zum Abgleichen von Begriffen in unstrukturierten Protokollereignissen](#)
- [Verwenden von Filtermustern zum Abgleichen von Begriffen in JSON-Protokollereignissen](#)
- [Verwenden von Filtermustern, um Begriffe in Leerzeichen-getrennten Protokollereignissen abzugleichen](#)

Unterstützte Syntax für reguläre Ausdrücke (Regex)

Unterstützte Regex-Syntax

Wenn Sie Regex zum Suchen und Filtern von Protokolldaten verwenden, müssen Sie Ihre Ausdrücke mit % umgeben.

Filtermuster mit Regex können nur Folgendes enthalten:

- Alphanumerische Zeichen – Ein alphanumerisches Zeichen ist ein Zeichen, das entweder ein Buchstabe (von A bis Z oder A bis Z) oder eine Ziffer (von 0 bis 9) ist.
- Unterstützte Symbolzeichen — Dazu gehören: ':', '_', '#', '=', '@', '/', ';', ',', '" und - "%something! % würde beispielsweise abgelehnt werden, da '!' nicht unterstützt wird.
- Unterstützte Operatoren – Dazu gehören: '^', '\$', '?', '[', ']', '{', '}', '|', '\\', '*', '+' und '.'.

Die Operatoren (und) werden nicht unterstützt. Sie dürfen keine Klammern verwenden, um ein Untermuster zu definieren.

Multi-Byte-Zeichen werden nicht unterstützt.

Note

Kontingente

Bei der Erstellung von metrischen Filtern oder Abonnementfiltern können Sie für jede Protokollgruppe maximal 5 Filtermuster mit Regex verwenden.

Es gibt ein Limit von 2 Regex für jedes Filtermuster, wenn Sie ein abgegrenztes oder JSON-Filtermuster für metrische Filter und Abonnementfilter erstellen oder wenn Sie Protokollereignisse oder Live Tail filtern.

Verwendung unterstützter Operatoren

- ^: Verankert die Übereinstimmung am Anfang einer Zeichenfolge. Beispielsweise stimmt %[hc]at% mit „hat“ und „cat“ überein, aber nur am Anfang einer Zeichenfolge.
- \$: Verankert die Übereinstimmung am Ende einer Zeichenfolge. Beispielsweise stimmt %[hc]at\$% mit „hat“ und „cat“ überein, aber nur am Ende einer Zeichenfolge.
- ?: Entspricht null oder einem Vorkommen des vorhergehenden Begriffs. So kann %colou?r% sowohl mit „color“ als auch „colour“ übereinstimmen.

- `[]`: Definiert eine Zeichenklasse. Gleicht die Zeichenliste oder den Zeichenbereich innerhalb der Klammern ab. Zum Beispiel stimmt `%[abc]%` mit „a“, „b“ oder „c“ überein; `%[a-z]%` stimmt mit jedem Kleinbuchstaben von „a“ bis „z“ überein; und `%[abcx-z]%` stimmt mit „a“, „b“, „c“, „x“, „y“ oder „z“ überein.
- `{m, n}`: Gleicht den vorhergehenden Begriff mindestens mit m-mal ab, aber nicht öfter als n-mal. `%a{3,5}%` stimmt beispielsweise nur mit „aaa“, „aaaa“ und „aaaaa“ überein.

 Note

Entweder m oder n können weggelassen werden, wenn Sie kein Minimum oder Maximum definieren möchten.

- `|`: Boolesches „Oder“, das den Begriff auf beiden Seiten des senkrechten Balkens abgleicht. Beispiel:
 - `%gray|ey%` kann „gray“ oder „grey“ entsprechen
 - `%^starting|^initializing|^shutting down%` kann mit „Start...“ übereinstimmen „oder“, „wird initialisiert...“ oder „Herunterfahren“, passt aber nicht zu „Initialisierung überspringen...“
 - `%abcc|ab[^c]$%` kann mit „abcc...“ und „aba...“ übereinstimmen, aber nicht mit „aac...“
- `\`: Escape-Zeichen, mit dem Sie die wörtliche Bedeutung eines Operators anstelle seiner speziellen Bedeutung verwenden können. Zum Beispiel stimmt `%\[.\\]%` mit jedem einzelnen Zeichen überein, das von `[` und `]` umgeben ist, da die Klammern ausgestellt sind, wie `[a]`, `[b]`, `[7]`, `[@]`, `[]` und `[ ]`.

 Note

`%10\.10\.0\.1%` ist die richtige Methode, um eine Regex zu erstellen, die der IP-Adresse 10.10.0.1 entspricht.

- `*`: Entspricht null oder mehr Instanzen des vorhergehenden Begriffs. So kann `%ab*c%` beispielsweise mit „ac“, „abc“ und „abbbc“ übereinstimmen; `%ab[0-9]*%` kann mit „ab“, „ab0“ und „ab129“ übereinstimmen.
- `+`: Entspricht einer oder mehreren Instanzen des vorhergehenden Begriffs. `%ab+c%` kann beispielsweise mit „abc“, „abbc“ und „abbbc“ übereinstimmen, aber nicht mit „ac“.
- `.`: Stimmt mit einem beliebigen Einzelzeichen überein. Zum Beispiel stimmt `%.at%` mit jeder Zeichenfolge mit drei Zeichen überein, die mit „at“ endet, einschließlich „hat“, „cat“, „bat“, „4at“, „#at“ und „ at“ (beginnend mit einem Leerzeichen).

 Note

Wenn Sie eine Regex erstellen, die IP-Adressen abgleicht, ist es wichtig, den Operator `.` auszustellen. Beispielsweise kann `%10.10.0.1%` mit „10010,051“ übereinstimmen, was möglicherweise nicht der eigentliche Verwendungszweck des Ausdrucks ist.

- `\d`, `\D`: Stimmt mit einer Ziffer bzw. einem Zeichen überein, das keine Ziffer ist. Beispielsweise stimmt `%\d%` mit `%[0-9]%` überein und `%\D%` mit `%[^0-9]%`.

 Note

Der Operator in Großbuchstaben bezeichnet die Umkehrung seines Gegenstücks in Kleinbuchstaben.

- `\s`, `\S`: Stimmt mit einem Leerraumzeichen/einem Nicht-Leerraumzeichen überein.

 Note

Der Operator in Großbuchstaben bezeichnet die Umkehrung seines Gegenstücks in Kleinbuchstaben. Zu den Leerraumzeichen gehören Tabulatorzeichen (`\t`), Leerzeichen () und Zeilenumbruch (`\n`).

- `\w`, `\W`: Stimmt mit einem alphanumerischen Zeichen/einem nicht-alphanumerischen Zeichen überein. Beispielsweise stimmt `%\w%` mit `%[a-zA-Z_0-9]%` überein und `%\W%` mit `%[^a-zA-Z_0-9]%`.

 Note

Der Operator in Großbuchstaben bezeichnet die Umkehrung seines Gegenstücks in Kleinbuchstaben.

- `\xhh`: Stimmt mit der ASCII-Zuordnung für ein zweistelliges hexadezimalen Zeichen überein. `\x` ist die Escape-Sequenz, die angibt, dass die folgenden Zeichen den hexadezimalen Wert für ASCII darstellen. `hh` gibt die beiden hexadezimalen Ziffern (0–9 und A–F) an, die auf ein Zeichen in der ASCII-Tabelle zeigen.

 Note

Sie können \xhh verwenden, um nach Symbolzeichen zu suchen, die vom Filtermuster nicht unterstützt werden. Beispielsweise stimmt %\x3A% mit : überein und %\x28% mit (.

Verwenden von Filtermustern zum Abgleichen von Begriffen mit einem regulären Ausdruck (Regex)

Begriffe mithilfe von Regex abgleichen

Sie können Begriffe in Ihren Protokollereignissen mit einem Regex-Muster abgleichen, das von % (Prozentzeichen vor und nach dem Regex-Muster) umgeben ist. Der folgende Codeauszug zeigt ein Beispiel für ein Filtermuster, das alle Protokollereignisse mit dem Schlüsselwort AUTHORIZED zurückgibt.

Eine Liste der unterstützten regulären Ausdrücke finden Sie unter [Unterstützte reguläre Ausdrücke](#).

```
%AUTHORIZED%
```

Dieses Filtermuster gibt Protokollereignismeldungen wie die folgenden zurück:

- [ERROR 401] UNAUTHORIZED REQUEST
- [SUCCESS 200] AUTHORIZED REQUEST

Verwenden von Filtermustern zum Abgleichen von Begriffen in unstrukturierten Protokollereignissen

Begriffe in unstrukturierten Protokollereignissen abgleichen

Die folgenden Beispiele enthalten Codeauszüge, die zeigen, wie Sie Filtermuster verwenden können, um Begriffe in Ihren unstrukturierten Protokollereignissen abzugleichen.

 Note

Filtermuster beachten die Groß-/Kleinschreibung. Schließen Sie exakte Phrasen und Begriffe, die nicht alphanumerische Zeichen enthalten, in doppelte Anführungszeichen ("") ein.

Example: Match a single term

Das folgende Code-Snippet zeigt ein Beispiel für ein begriffsspezifisches Filtermuster, das alle Protokollereignisse zurückgibt, deren Meldungen das Wort ERROR (FEHLER) enthalten.

```
ERROR
```

Das Filtermuster gleicht Protokollereignismeldungen ab, z. B. die folgende:

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST
- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

Example: Match multiple terms

Das folgende Code-Snippet zeigt ein Beispiel für ein Filtermuster für mehrere Begriffe, das alle Protokollereignisse zurückgibt, deren Meldungen die Wörter ERROR (FEHLER) und ARGUMENTS (ARGUMENTE) enthalten.

```
ERROR ARGUMENTS
```

Der Filter gibt Protokollereignismeldungen zurück, wie z. B. die folgende:

- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

Dieses Filtermuster gibt die folgenden Protokollereignismeldungen nicht zurück, da sie nicht beide im Filtermuster angegebenen Begriffe enthalten.

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST

Example: Match optional terms

Mithilfe des Musterabgleichs können Sie Filtermuster erstellen, die Protokollereignisse mit optionalen Begriffen zurückgeben. Setzen Sie ein Fragezeichen („?“) vor die Begriffe, die Sie abgleichen möchten. Das folgende Code-Snippet zeigt ein Beispiel für ein Filtermuster, das alle Protokollereignisse zurückgibt, deren Meldungen das Wort ERROR oder das Wort ARGUMENTS enthalten.

```
?ERROR ?ARGUMENTS
```

Das Filtermuster gleicht Protokollereignismeldungen ab, z. B. die folgende:

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST
- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

Note

Sie können das Fragezeichen („?“) nicht kombinieren mit anderen Filtermustern, z. B. Begriffe einschließen und ausschließen. Wenn Sie „?“ kombinieren Bei anderen Filtermustern werden alle Begriffe mit Fragezeichen ignoriert.

Das folgende Filtermuster entspricht beispielsweise allen Ereignissen, die das Wort REQUEST, aber das Fragezeichen („?“) enthalten Filterbegriffe werden ignoriert und haben keine Wirkung.

```
?ERROR ?ARGUMENTS REQUEST
```


Übereinstimmungen von Protokollereignissen

- [INFO] REQUEST FAILED
- [WARN] UNAUTHORIZED REQUEST
- [ERROR] 400 BAD REQUEST

Example: Match exact phrases

Das folgende Code-Snippet zeigt ein Beispielfiltermuster, das Protokollereignisse zurückgibt, deren Meldungen genau den Ausdruck INTERNAL SERVER ERROR (INTERNER SERVERFEHLER) enthalten.

```
"INTERNAL SERVER ERROR"
```

Dieses Filtermuster gibt die folgende Ereignisprotokollmeldung zurück:

- [ERROR 500] INTERNAL SERVER ERROR

Example: Include and exclude terms

Sie können Filtermuster erstellen, die Protokollereignisse zurückgeben, bei denen Meldungen bestimmte Begriffe enthalten und andere ausschließen. Setzen Sie ein Minuszeichen („-“) vor die Begriffe, die Sie ausschließen möchten. Das folgende Code-Snippet zeigt ein Beispielfiltermuster, das Protokollereignisse zurückgibt, deren Meldungen den Begriff ERROR (FEHLER) enthalten, nicht aber den Begriff ARGUMENTS (ARGUMENTE).

```
ERROR -ARGUMENTS
```

Dieses Filtermuster gibt Protokollereignismeldungen wie die folgenden zurück:

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST

Dieses Filtermuster gibt die folgenden Protokollereignismeldungen nicht zurück, da sie den Begriff ARGUMENTS (ARGUMENTE) enthalten.

- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

Example: Match everything

Mit doppelten Anführungszeichen können Sie alles in Ihren Protokollereignissen finden. Das folgende Code-Snippet zeigt ein Beispiel für ein Filtermuster, das alle Protokollereignisse zurückgibt.

```
" "
```

Verwenden von Filtermustern zum Abgleichen von Begriffen in JSON-Protokollereignissen

Schreiben von Filtermustern für JSON-Protokollereignisse

Im Folgenden wird beschrieben, wie Sie die Syntax für Filtermuster schreiben, die mit JSON-Begriffen übereinstimmen, die Zeichenfolgen und numerische Werte enthalten.

Writing filter patterns that match strings

Sie können Filtermuster erstellen, um Zeichenfolgen in JSON-Protokollereignissen abzugleichen. Der folgende Codeauszug zeigt ein Syntaxbeispiel für zeichenfolgenbasierte Filtermuster.

```
{ PropertySelector EqualityOperator String }
```

Schließen Sie Filtermuster in geschweifte Klammern („{}“) ein. Zeichenfolgenbasierte Filtermuster müssen die folgenden Elemente enthalten:

- Eigenschaftsselektor

Kennzeichnen Sie Eigenschaftsselektoren mit einem Dollarzeichen gefolgt von einem Punkt („\$.“). Eigenschaftsselektoren sind alphanumerische Zeichenfolgen, die Bindestriche („-“) und Unterstriche („_“) unterstützen. Zeichenketten unterstützen keine wissenschaftliche Notation. Eigenschaftsselektoren verweisen auf Werteknoten in JSON-Protokollereignissen. Werteknoten können Zeichenketten oder Zahlen sein. Platzieren Sie Arrays nach Eigenschaftsselektoren. Für die Elemente in Arrays wird ein nullbasiertes Nummerierungssystem verwendet. Somit ist das erste Element im Array das Element 0, das zweite Element ist das Element 1 und so weiter. Schließen Sie Elemente in Klammern ein („[]“). Wenn ein Eigenschaftsselektor auf ein Array oder Objekt verweist, stimmt das Filtermuster nicht mit dem Protokollformat überein. Wenn die JSON-Eigenschaft einen Punkt („.“) enthält, kann die Klammernotation verwendet werden, um diese Eigenschaft auszuwählen.



Note

Platzhalter-Selektor

Sie können den JSON-Platzhalter verwenden, um ein beliebiges Array-Element oder ein beliebiges JSON-Objektfeld auszuwählen.

Kontingente

Sie können in einem Eigenschaftenselektor nur bis zu einem Platzhalter-Selektor verwenden.

- Equality operator (Gleichheitsoperator)

Kennzeichnen Sie Gleichheitsoperatoren mit einem der folgenden Symbole: gleich („=“) oder ungleich („!=“). Gleichheitsoperatoren geben einen booleschen Wert zurück („true“ oder „false“).

- Zeichenfolge

Sie können Zeichenketten in doppelte Anführungszeichen („“) einschließen. Zeichenketten, die andere Typen als alphanumerische Zeichen und das Unterstrichsymbol enthalten, müssen in doppelte Anführungszeichen gesetzt werden. Verwenden Sie das Sternchen („*“) als Platzhalter, um Text abzugleichen.

 Note

Sie können jeden bedingten regulären Ausdruck verwenden, wenn Sie Filtermuster erstellen, um Begriffe in JSON-Protokollereignissen abzugleichen. Eine Liste der unterstützten regulären Ausdrücke finden Sie unter [Unterstützte reguläre Ausdrücke](#).

Der folgende Codeauszug enthält ein Beispiel für ein Filtermuster. Es zeigt, wie Sie ein Filtermuster so formatieren können, dass es einen JSON-Begriff mit einer Zeichenfolge abgleicht.

```
{ $.eventType = "UpdateTrail" }
```

Writing filter patterns that match numeric values

Sie können Filtermuster erstellen, um numerische Werte in JSON-Protokollereignissen abzugleichen. Der folgende Codeauszug zeigt ein Beispiel für die Syntax von Filtermustern, die numerische Werte abgleichen.

```
{ PropertySelector NumericOperator Number }
```

Schließen Sie Filtermuster in geschweifte Klammern („{ }“) ein. Filtermuster, die numerische Werten abgleichen, müssen die folgenden Elemente enthalten:

- Eigenschaftsselektor

Kennzeichnen Sie Eigenschaftsselektoren mit einem Dollarzeichen gefolgt von einem Punkt („\$.“). Eigenschaftsselektoren sind alphanumerische Zeichenfolgen, die Bindestriche („-“) und Unterstriche („_“) unterstützen. Zeichenketten unterstützen keine wissenschaftliche Notation. Eigenschaftsselektoren verweisen auf Werteknoten in JSON-Protokollereignissen. Werteknoten können Zeichenketten oder Zahlen sein. Platzieren Sie Arrays nach Eigenschaftsselektoren. Für die Elemente in Arrays wird ein nullbasiertes Nummerierungssystem verwendet. Somit ist das erste Element im Array das Element 0, das zweite Element ist das Element 1 und so weiter. Schließen Sie Elemente in Klammern ein („[]“). Wenn ein Eigenschaftsselektor auf ein Array oder Objekt verweist, stimmt das Filtermuster nicht mit dem Protokollformat überein. Wenn die

JSON-Eigenschaft einen Punkt (".") enthält, kann die Klammernotation verwendet werden, um diese Eigenschaft auszuwählen.

 Note

Platzhalter-Selektor

Sie können den JSON-Platzhalter verwenden, um ein beliebiges Array-Element oder ein beliebiges JSON-Objektfeld auszuwählen.

Kontingente

Sie können in einem Eigenschaftenselektor nur bis zu einen Platzhalter-Selektor verwenden.

- Numerischer Operator

Kennzeichnen Sie numerische Operatoren mit einem der folgenden Symbole: größer als („>“), kleiner als („<“), gleich („=“), ungleich („!=“), größer als oder gleich („>=“), oder kleiner als oder gleich („<=“).

- Zahl

Sie können Ganzzahlen verwenden, die Plus- („+“) oder Minuszeichen („-“) enthalten und der wissenschaftlichen Notation folgen. Verwenden Sie das Sternchen („*“) als Platzhalter, um Zahlen abzugleichen.

Der folgende Codeauszug enthält Beispiele dafür, wie Sie Formatfiltermuster so formatieren können, dass sie JSON-Begriffe mit numerischen Werten abgleichen.

```
// Filter pattern with greater than symbol
{ $.bandwidth > 75 }
// Filter pattern with less than symbol
{ $.latency < 50 }
// Filter pattern with greater than or equal to symbol
{ $.refreshRate >= 60 }
// Filter pattern with less than or equal to symbol
{ $.responseTime <= 5 }
// Filter pattern with equal sign
{ $.errorCode = 400}
// Filter pattern with not equal sign
{ $.errorCode != 500 }
// Filter pattern with scientific notation and plus symbol
```

```
{ $.number[0] = 1e+3 }  
// Filter pattern with scientific notation and minus symbol  
{ $.number[0] != 1e-3 }
```

Begriffe in JSON-Protokollereignissen mit einfachen Ausdrücken abgleichen

Die folgenden Beispiele enthalten Codeauszüge, die zeigen, wie Filtermuster Begriffe in einem JSON-Protokollereignis abgleichen können.

Note

Wenn Sie die Beispielfiltermuster mit dem Beispiel-JSON-Protokollereignis testen, müssen Sie das Beispiel-JSON-Protokoll in einer einzigen Zeile eingeben.

JSON-Protokollereignis

```
{  
  "eventType": "UpdateTrail",  
  "sourceIPAddress": "111.111.111.111",  
  "arrayKey": [  
    "value",  
    "another value"  
  ],  
  "objectList": [  
    {  
      "name": "a",  
      "id": 1  
    },  
    {  
      "name": "b",  
      "id": 2  
    }  
  ],  
  "SomeObject": null,  
  "cluster.name": "c"  
}
```

Example: Filter pattern that matches string values

Dieses Filtermuster stimmt mit der Zeichenfolge "UpdateTrail" in der Eigenschaft "eventType" überein.

```
{ $.eventType = "UpdateTrail" }
```

Example: Filter pattern that matches string values (IP address)

Dieses Filtermuster enthält einen Platzhalter und stimmt mit der Eigenschaft "sourceIPAddress" überein, da sie keine Zahl mit dem Präfix "123.123." enthält.

```
{ $.sourceIPAddress != 123.123.* }
```

Example: Filter pattern that matches a specific array element with a string value

Dieses Filtermuster stimmt mit dem Element "value" im Array "arrayKey" überein.

```
{ $.arrayKey[0] = "value" }
```

Example: Filter pattern that matches a string using regex

Dieses Filtermuster stimmt mit der Zeichenfolge "Trail" in der Eigenschaft "eventType" überein.

```
{ $.eventType = %Trail% }
```

Example: Filter pattern that uses a wildcard to match values of any element in the array using regex

Das Filtermuster enthält Regex, der mit dem Element "value" im Array "arrayKey" übereinstimmt.

```
{ $.arrayKey[*] = %val.{2}% }
```

Example: Filter pattern that uses a wildcard to match values of any element with a specific prefix and subnet using regex (IP address)

Dieses Filtermuster enthält Regex, der mit dem Element "111.111.111.111" in der Eigenschaft "sourceIPAddress" übereinstimmt.

```
{ $.* = %111\.111\.111\.1[0-9]{1,2}% }
```

 Note

Kontingente

Sie können in einem Eigenschaftenselektor nur bis zu einen Platzhalter-Selektor verwenden.

Example: Filter pattern that matches a JSON property with a period (.) in the key

```
{ $.['cluster.name'] = "c" }
```

Example: Filter pattern that matches JSON logs using IS

Sie können Filtermuster erstellen, die Felder in JSON-Protokollen mit der Variable IS abgleichen. Die Variable IS kann Felder abgleichen, die die Werte NULL, TRUE oder FALSE enthalten. Das folgende Filtermuster gibt JSON-Protokolle zurück, bei denen der Wert von SomeObject NULL ist.

```
{ $.SomeObject IS NULL }
```

Example: Filter pattern that matches JSON logs using NOT EXISTS

Sie können mit der NOT EXISTS Variablen Filtermuster erstellen, um JSON-Logs zurückzugeben, die keine bestimmten Felder in den Protokolldaten enthalten. Das folgende Filtermuster verwendet

NOT EXISTS, um JSON-Protokolle zurückzugeben, die das Feld `SomeOtherObject` nicht enthalten.

```
{ $.SomeOtherObject NOT EXISTS }
```

Note

Die Variablen `IS NOT` und `EXISTS` werden derzeit nicht unterstützt.

Begriffe in JSON-Objekten mit zusammengesetzten Ausdrücken abgleichen

Sie können die logischen Operatoren AND („&&“) und OR („||“) in Filtermustern verwenden, um zusammengesetzte Ausdrücke zu erstellen, die Protokollereignisse abgleichen, bei denen zwei oder mehr Bedingungen erfüllt sind. Zusammengesetzte Ausdrücke unterstützen die Verwendung von Klammern („()“) und die folgende Standardreihenfolge von Operationen: `() > && > ||`. Die folgenden Beispiele enthalten Codeauszüge, die zeigen, wie Sie Filtermuster mit zusammengesetzten Ausdrücken verwenden können, um Begriffe in einem JSON-Objekt abzugleichen.

JSON-Objekt

```
{
  "user": {
    "id": 1,
    "email": "John.Stiles@example.com"
  },
  "users": [
    {
      "id": 2,
      "email": "John.Doe@example.com"
    },
    {
      "id": 3,
      "email": "Jane.Doe@example.com"
    }
  ],
  "actions": [
    "GET",
    "PUT",
```

```
    "DELETE"
  ],
  "coordinates": [
    [0, 1, 2],
    [4, 5, 6],
    [7, 8, 9]
  ]
}
```

Example: Expression that matches using AND (&&)

Das Filtermuster enthält einen zusammengesetzten Ausdruck, der "id" in "user" mit einem numerischen Wert von 1 und "email" im ersten Element des Arrays "users" mit der Zeichenfolge "John.Doe@example.com" abgleicht.

```
{ ($.user.id = 1) && ($.users[0].email = "John.Doe@example.com") }
```

Example: Expression that matches using OR (||)

Dieses Filtermuster enthält einen zusammengesetzten Ausdruck, der "email" in "user" mit der Zeichenfolge "John.Stiles@example.com" abgleicht.

```
{ $.user.email = "John.Stiles@example.com" || $.coordinates[0][1] = "nonmatch" &&
  $.actions[2] = "nonmatch" }
```

Example: Expression that doesn't match using AND (&&)

Dieses Filtermuster enthält einen zusammengesetzten Ausdruck, der keine Übereinstimmung findet, weil der Ausdruck nicht mit der dritten Aktion in "actions" übereinstimmt.

```
{ ($.user.email = "John.Stiles@example.com" || $.coordinates[0][1] = "nonmatch") &&
  $.actions[2] = "nonmatch" }
```

 Note

Kontingente

Sie können in einer Eigenschaftenauswahl nur bis zu einen Platzhalter-Selektor und bis zu drei Platzhalter-Selektoren in einem Filtermuster mit zusammengesetzten Ausdrücken verwenden.

Example: Expression that doesn't match using OR (||)

Dieses Filtermuster enthält einen zusammengesetzten Ausdruck, der keine Übereinstimmung findet, weil der Ausdruck nicht mit der ersten Eigenschaft in "users" oder der dritten Aktion in "actions" übereinstimmt.

```
{ ($.user.id = 2 && $.users[0].email = "nonmatch") || $.actions[2] = "GET" }
```

Verwenden von Filtermustern, um Begriffe in Leerzeichen-getrennten Protokollereignissen abzugleichen

Schreiben von Filtermustern für Leerzeichen-getrennte Protokollereignisse

Sie können Filtermuster verwenden, um Begriffe in Leerzeichen-getrennten Protokollereignissen abzugleichen. Im Folgenden finden Sie ein Beispiel für ein durch Leerzeichen getrenntes Protokollereignis und es wird beschrieben, wie die Syntax für Filtermuster geschrieben wird, die Begriffe im Leerzeichen-getrennten Protokollereignis abgleichen.

 Note

Sie können jeden bedingten regulären Ausdruck verwenden, wenn Sie Filtermuster erstellen, um Begriffe in Leerzeichen-getrennten Protokollereignissen abzugleichen. Eine Liste der unterstützten regulären Ausdrücke finden Sie unter [Unterstützte reguläre Ausdrücke](#).

Example: Space-delimited log event

Das folgende Code-Snippet zeigt ein Leerzeichen-getrenntes Protokollereignis, das sieben Felder enthält: `ip`, `user`, `username`, `timestamp`, `request`, `status_code` und `bytes`.

```
127.0.0.1 Prod frank [10/Oct/2000:13:25:15 -0700] "GET /index.html HTTP/1.0" 404 1534
```

Note

Zeichen zwischen Klammern („[]“) und doppelten Anführungszeichen („“) gelten als einzelne Felder.

Writing filter patterns that match terms in a space-delimited log event

Um ein Filtermuster zu erstellen, das mit Begriffen in einem durch Leerzeichen getrennten Protokollereignis übereinstimmt, schließen Sie das Filtermuster in eckige Klammern („[]“) ein und geben Felder mit Namen an, die durch Kommas („“,“) getrennt sind. Das folgende Filtermuster analysiert sieben Felder.

```
[ip=%127\.\0\.\0\[1-9]%, user, username, timestamp, request =*.html*, status_code = 4*, bytes]
```

Sie können numerische Operatoren (`>`, `<`, `=`, `!=`, `>=` oder `<=`) sowie das Sternchen (`*`) als Platzhalter oder Regex verwenden, um die Filtermusterbedingungen anzugeben. Im Beispielfiltermuster verwendet `ip` einen Regex, der auf den IP-Adressbereich 127.0.0.1–127.0.0.9 abgleicht, `request` enthält einen Platzhalter, der besagt, dass ein Wert mit `.html` extrahiert werden muss, und `status_code` enthält einen Platzhalter, der besagt, dass ein Wert mit 4 beginnen muss.

Wenn Sie die Anzahl der Felder, die Sie in einem Leerzeichen-getrennten Protokollereignis analysieren, nicht kennen, können Sie mit Ellipsen (...) auf jedes unbenannte Feld verweisen. Mit Ellipsen können Sie auf so viele Felder wie nötig verweisen. Das folgende Beispiel zeigt ein Filtermuster mit Ellipsen, die die ersten vier unbenannten Felder aus dem vorherigen Beispielfiltermuster darstellen.

```
[..., request =*.html*, status_code = 4*, bytes]
```

Sie können auch die logischen Operatoren AND (&&) und OR (||) verwenden, um zusammengesetzte Ausdrücke zu erstellen. Das folgende Filtermuster enthält einen zusammengesetzten Ausdruck, der besagt, dass der Wert von `status_code` entweder 404 oder 410 sein muss.

```
[ip, user, username, timestamp, request =*.html*, status_code = 404 || status_code = 410, bytes]
```

Begriffe in Leerzeichen-getrennten Protokollereignissen mit Musterabgleich abgleichen

Sie können den Musterabgleich verwenden, um Leerzeichen-getrennte Filtermuster zu erstellen, die Begriffe in einer bestimmten Reihenfolge abgleichen. Geben Sie die Reihenfolge Ihrer Begriffe mit Indikatoren an. Verwenden Sie `w1` für den ersten Begriff, `w2` für den zweiten und so weiter, um die Reihenfolge der Begriffe abzubilden. Platzieren Sie Kommas (,,") zwischen Ihren Begriffen. Die folgenden Beispiele enthalten Codeauszüge, die zeigen, wie Sie den Musterabgleich mit Leerzeichen-getrennten Filtermustern verwenden können.

Note

Sie können jeden bedingten regulären Ausdruck verwenden, wenn Sie Filtermuster erstellen, um Begriffe in Leerzeichen-getrennten Protokollereignissen abzugleichen. Eine Liste der unterstützten regulären Ausdrücke finden Sie unter [Unterstützte reguläre Ausdrücke](#).

Beispiel: Leerzeichen-getrenntes Protokollereignis

```
INFO 09/25/2014 12:00:00 GET /service/resource/67 1200
INFO 09/25/2014 12:00:01 POST /service/resource/67/part/111 1310
WARNING 09/25/2014 12:00:02 Invalid user request
ERROR 09/25/2014 12:00:02 Failed to process request
```

Example: Match terms in order

Das folgende Leerzeichen-getrennte Filtermuster gibt Protokollereignisse zurück, bei denen das erste Wort in den Protokollereignissen ERROR (FEHLER) lautet.

```
[w1=ERROR, w2]
```

Note

Wenn Sie Leerzeichen-getrennte Filtermuster erstellen, die Musterabgleiche verwenden, müssen Sie nach der Angabe der Reihenfolge der Begriffe einen leeren Indikator einfügen. Wenn Sie beispielsweise ein Filtermuster erstellen, der Protokollereignisse zurückgibt, deren erstes Wort ERROR (FEHLER) lautet, fügen Sie einen leeren w2-Indikator nach dem w1-Begriff ein.

Example: Match terms with AND (&&) and OR (||)

Sie können die logischen Operatoren AND („&&“) und OR („||“) verwenden, um Leerzeichen-getrennte Filtermuster zu erstellen, die Bedingungen enthalten. Das folgende Filtermuster gibt Protokollereignisse zurück, bei denen der erste Begriff in den Ereignissen ERROR (FEHLER) oder WARNING (WARNUNG) lautet.

```
[w1=ERROR || w1=WARNING, w2]
```

Example: Exclude terms from matches

Sie können Leerzeichen-getrennte Filtermuster erstellen, die Protokollereignisse zurückgeben, die einen oder mehrere Begriffe ausschließen. Setzen Sie ein Ungleichheitszeichen („!=“) vor den Begriff oder die Begriffe, die Sie ausschließen möchten. Der folgende Codeauszug zeigt ein

Beispiel für ein Filtermuster, das Protokollereignisse zurückgibt, deren erste Begriffe nicht ERROR (FEHLER) und WARNING (WARNUNG) lauten.

```
[w1!=ERROR && w1!=WARNING, w2]
```

Example: Match the top level item in a resource URI

Der folgende Codeauszug zeigt ein Beispiel für ein Filtermuster, das das Element der obersten Ebene in einer Ressourcen-URI mit Regex abgleicht.

```
[logLevel, date, time, method, url=%/service/resource/[0-9]+$, response_time]
```

Example: Match the child level item in a resource URI

Der folgende Codeauszug zeigt ein Beispiel für ein Filtermuster, das das Element der untergeordneten Ebene in einer Ressourcen-URI mit Regex abgleicht.

```
[logLevel, date, time, method, url=%/service/resource/[0-9]+/part/[0-9]+$,  
response_time]
```

Protokollierung aktivieren von AWS service

Während viele Dienste Protokolle nur in CloudWatch Logs veröffentlichen, können einige AWS Services Protokolle direkt in Amazon Simple Storage Service oder Amazon Data Firehose veröffentlichen. Wenn Ihre Hauptanforderung an Protokolle die Speicherung oder Verarbeitung in einem dieser Dienste ist, können Sie den Dienst, der die Protokolle erstellt, diese ohne zusätzliche Einrichtung direkt an Amazon S3 oder Firehose senden lassen.

Auch wenn Sie Protokolle direkt auf Amazon S3 oder Firehose veröffentlichen, fallen CloudWatch Versandkosten an. Wenn Sie Protokolle an Amazon S3 senden, erscheinen die **AWS_REGION-S3-Egress-Bytes** Gebühren im Cost Explorer oder auf Ihrer Rechnung. Wenn Sie Protokolle an Firehose senden, **AWS_REGION-FH-Egress-Bytes** fallen Gebühren an. Weitere Informationen zu den Preisen für verkaufte Logs finden Sie auf der Registerkarte Logs unter [Amazon CloudWatch Pricing](#).

Einige AWS Dienste verwenden eine gemeinsame Infrastruktur, um ihre Protokolle zu versenden. Um die Protokollierung dieser Services zu aktivieren, müssen Sie als Benutzer angemeldet sein, der über bestimmte Berechtigungen verfügt. Darüber hinaus müssen Sie Berechtigungen erteilen, AWS um das Senden der Protokolle zu ermöglichen.

Für Dienste, die diese Berechtigungen benötigen, gibt es zwei Versionen der benötigten Berechtigungen. Die Dienste, für die diese zusätzlichen Berechtigungen erforderlich sind, sind in der als Unterstützte [V1-Berechtigungen] und Unterstützte [V2-Berechtigungen] aufgeführt [the section called “Unterstützte Protokollziele”](#). Informationen zu diesen erforderlichen Berechtigungen finden Sie in den Abschnitten nach der Tabelle.

<integration-catalog>

<integration></integration>

<integration></integration>

<integration></integration>

<integration></integration>

<integration></integration>

<integration></integration>

<integration></integration>

<integration></integration>

<integration></integration>

<integration></integration>

<integration></integration>

[illegible]

[illegible]

Unterstützte Protokollziele

Die folgende Tabelle zeigt, welche Ziele jeder AWS Dienst für das Senden von Protokollen unterstützt und welche Berechtigungsversion erforderlich ist.

Protokollquelle	Protokoll typ	the section called “An Logs gesendete CloudWatc h Protokoll e”	the section called “An Amazon S3 gesendete Protokoll e”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
Amazon API Gateway Gateway-Zugriffsprotokolle	Vended-Protokolle	Unterstützte [V1 Berechtigungen]			
AWS AppSync logs	Custom logs (Benutzer definierte Protokolle)	Unterstützt			
Amazon Aurora MySQL-Protokolle	Custom logs (Benutzer definierte Protokolle)	Unterstützt			
Amazon Bedrock Protokollierung in Wissensdatenbanken	Vended-Protokolle	Unterstützte [V2]	Unterstützte [V2]	Unterstützte [V2]	

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
		Berechtigungen]	Berechtigungen]	Berechtigungen]	
Amazon Bedrock Protokollierung durch Agenten	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
Amazon Bedrock AgentCore Laufzeit	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]
Amazon Bedrock AgentCore Gateway	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]
Amazon Bedrock AgentCore Identität	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]
Amazon Bedrock AgentCore Arbeitsspeicher	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
Amazon Bedrock AgentCore Zahlungen	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]
Amazon Bedrock AgentCore Tools	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]
Amazon Chime Metrikprotokolle zur Medienqualität und SIP-Nachrichtenprotokolle	Vended-Protokolle	Unterstützte [V1 Berechtigungen]			
CloudFront: Zugriffsprotokolle	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
AWS CloudHSM Audit-Protokolle	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
CloudWatch Offensichtlich Evaluierungseignisprotokolle	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]		
CloudWatch Internet Monitor-Protokolle	Vended-Protokolle		Unterstützte [V1 Berechtigungen]		
CloudTrail logs	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			
AWS CodeBuild logs	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
Amazon CodeWhisperer Ereignisprotokolle	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
Amazon Cognito logs	Vended-Protokolle	Unterstützte [V1 Berechtigungen]			
Amazon Connect Kundenprotokolle	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			
AWS DataSync logs	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
AWS DevOps Agent logs	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
ElastiCache Amazon-Protokolle (Redis OSS)	Vended-Protokolle	Unterstützte [V1 Berechtigungen]		Unterstützte [V1 Berechtigungen]	
AWS Elastic Beanstalk logs	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			
Amazon Elastic Container Service-Protokolle	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
Protokolle für den automatischen Modus von Amazon Elastic Kubernetes Service	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
Protokolle der Amazon Elastic Kubernetes Service Service-Kontrollebene	Vended-Protokolle	Unterstützt			
AWS Elemental MediaPackage Zugriffsprotokolle	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
AWS Elemental MediaTailor logs	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
AWS Entity Resolution logs	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
Amazon EventBridge Protokollierung von Rohrleitungen	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
Amazon EventBridge Busse für Veranstaltungen	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
AWS Fargate logs	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			
AWS Fault Injection Service Versuchsprotokolle	Vended-Protokolle		Unterstützte [V1 Berechtigungen]		
Amazon FinSpace	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	
AWS Global Accelerator Flussprotokolle	Vended-Protokolle		Unterstützte [V1 Berechtigungen]		

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
AWS Glue Auftragsprotokolle	Custom logs (Benutzer definierte Protokolle)	Unterstützt			
IAM Identity Center-Fehlerprotokolle	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
Chatprotokolle von Amazon Interactive Video Service	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	
AWS IoT logs	Custom logs (Benutzer definierte Protokolle)	Unterstützt			

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
AWS IoT FleetWise logs	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	
AWS Lambda logs	Vended-Protokolle	Unterstützt	Unterstützt	Unterstützt	
Amazon Macie Macie-Protokolle	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			
Amazon SES SES-Protokolle	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
AWS Mainframe Modernization	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
Protokolle von Amazon Managed Service für Prometheus	Vended-Protokolle	Unterstützte [V1 Berechtigungen]			
Amazon MSK-Brokerprotokolle	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	
Amazon MSK Connect-Protokolle	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	
Amazon MQ MQ-Protokolle	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			
AWS Netzwerk-Firewall-Protokolle	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
AWS Netzwerk-Firewall-Proxyprotokolle	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
Network Load Balancer Balancer-Zugriffsprotokolle	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
OpenSearch logs	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			
Amazon OpenSearch Service-Aufnahmeprotokolle	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	
AWS PCS logs	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
Amazon Q Business-Konnektorprotokolle	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
Gesprächsprotokolle von Amazon Q Business	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
Amazon Quick Chat- und Feedback-Protokolle	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
PostgreSQL-Protokolle von Amazon Relational Database Service	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			
AWS RTB Fabric-Protokolle	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokoll e”	the section called “An Amazon S3 gesendete Protokoll e”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
AWS Security Hub CSPM	Vended-Protokolle	Unterstützte [V2 Berechtigungen]			
AWS Security Hub	Vended-Protokolle	Unterstützte [V2 Berechtigungen]			
Öffentliche DNS-Abfrageprotokolle von Amazon Route 53	Vended-Protokolle	Unterstützt			
Amazon Route 53-Resolver-Abfrageprotokolle	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]		
Amazon SageMaker AI-Veranstaltungen	Vended-Protokolle	Unterstützte [V1 Berechtigungen]			
Veranstaltungen für Mitarbeiter von Amazon SageMaker AI	Vended-Protokolle	Unterstützte [V1 Berechtigungen]			

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
AWS Site-to-Site VPN-Protokolle	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	
Amazon Simple Email Service-Protokolle	Vended-Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	
Protokolle von Amazon Simple Notification Service	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			
Protokolle der Datenschutzrichtlinien von Amazon Simple Notification Service	Custom logs (Benutzerdefinierte Protokolle)	Unterstützt			

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokolle”	the section called “An Amazon S3 gesendete Protokolle”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
EC2-Spot-Instance-Datenfeed-Dateien	Vended-Protokolle		Unterstützte [V1 Berechtigungen]		
AWS Step Functions Express-Workflow- und Standard-Workflow-Protokolle	Vended-Protokolle	Unterstützte [V1 Berechtigungen]			
Storage Gateway Gateway-Auditprotokolle und Integritätsprotokolle	Vended-Protokolle	Unterstützte [V1 Berechtigungen]			
AWS Transfer Family logs	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	
AWS Verified Access logs	Vended-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	

Protokollquelle	Protokolltyp	the section called “An Logs gesendete CloudWatch h Protokoll e”	the section called “An Amazon S3 gesendete Protokoll e”	the section called “An Firehose gesendete Logs”	the section called “Spuren wurden gesendet an X-Ray”
Amazon Virtual Private Cloud Cloud-Flussprotokolle	Vended-Pr otokolle	Unterstüt zt	Unterstüt zte [V1 Berechtig ungen]	Unterstüt zte [V1 Berechtig ungen]	
Amazon VPC Lattice-Zugriffsprotokolle	Vended-Pr otokolle	Unterstüt zte [V1 Berechtig ungen]	Unterstüt zte [V1 Berechtig ungen]	Unterstüt zte [V1 Berechtig ungen]	
Amazon VPC-Routenserver	Vended-Pr otokolle	Unterstüt zte [V2 Berechtig ungen]	Unterstüt zte [V2 Berechtig ungen]	Unterstüt zte [V2 Berechtig ungen]	
AWS WAF logs	Vended-Pr otokolle	Unterstüt zte [V1 Berechtig ungen]	Unterstüt zte [V1 Berechtig ungen]	Unterstüt zt	
Amazon WorkMail Audit-Protokolle	Vended-Pr otokolle	Unterstüt zte [V2 Berechtig ungen]	Unterstüt zte [V2 Berechtig ungen]	Unterstüt zte [V2 Berechtig ungen]	

Protokollierung, für die zusätzliche Berechtigungen [V1] erforderlich sind

Einige AWS Dienste verwenden eine gemeinsame Infrastruktur, um ihre CloudWatch Protokolle an Logs, Amazon S3 oder Firehose zu senden. Damit die in der obigen Tabelle aufgeführten AWS Dienste ihre Protokolle an diese Ziele senden können, müssen Sie als Benutzer mit bestimmten Berechtigungen angemeldet sein.

Darüber hinaus müssen Berechtigungen erteilt werden, AWS damit die Protokolle gesendet werden können. AWS kann diese Berechtigungen automatisch erstellen, wenn die Protokolle eingerichtet werden, oder Sie können sie zuerst selbst erstellen, bevor Sie die Protokollierung einrichten. Für die kontoübergreifende Bereitstellung müssen Sie die Berechtigungsrichtlinien manuell selbst erstellen.

Wenn Sie sich dafür entscheiden, dass die erforderlichen Berechtigungen und Ressourcenrichtlinien AWS automatisch eingerichtet werden, wenn Sie oder jemand in Ihrer Organisation das Senden von Protokollen zum ersten Mal einrichten, muss der Benutzer, der das Senden von Protokollen einrichtet, über bestimmte Berechtigungen verfügen, wie später in diesem Abschnitt erläutert wird. Alternativ können Sie die Ressourcenrichtlinien selbst erstellen, und dann benötigen die Benutzer, die das Senden von Protokollen einrichten, nicht so viele Berechtigungen.

In den folgenden Themen finden Sie weitere Informationen zu den einzelnen Zielen.

Themen

- [An Logs gesendete CloudWatch Protokolle](#)
- [An Amazon S3 gesendete Protokolle](#)
- [An Firehose gesendete Logs](#)

An Logs gesendete CloudWatch Protokolle

Important

Wenn Sie die Protokolltypen in der folgenden Liste so einrichten, dass sie an CloudWatch Logs gesendet werden sollen, AWS erstellt oder ändert bei Bedarf die Ressourcenrichtlinien, die der Protokollgruppe zugeordnet sind, die die Protokolle empfängt. Lesen Sie diesen Abschnitt weiter, um mehr Details zu erhalten.

Dieser Abschnitt gilt, wenn die in der Tabelle im vorherigen Abschnitt aufgeführten Protokolltypen an CloudWatch Logs gesendet werden:

Benutzerberechtigungen

Um das erstmalige Senden dieser Protokolltypen an Logs einrichten zu CloudWatch können, müssen Sie bei einem Konto mit den folgenden Berechtigungen angemeldet sein.

- `logs:CreateLogDelivery`
- `logs:PutResourcePolicy`
- `logs:DescribeResourcePolicies`
- `logs:DescribeLogGroups`

Note

Wenn Sie die `logs:PutResourcePolicy` Berechtigung `logs:DescribeLogGroups``logs:DescribeResourcePolicies`, oder angeben, achten Sie darauf, den ARN der Resource Zeile so einzustellen, dass er einen * Platzhalter verwendet, anstatt nur einen einzigen Protokollgruppennamen anzugeben. Beispiel: "Resource": "arn:aws:logs:us-east-1:111122223333:log-group:*"

Wenn einer dieser Protokolltypen bereits an eine Protokollgruppe unter CloudWatch Logs gesendet wird, benötigen Sie nur die `logs:CreateLogDelivery` entsprechende Berechtigung, um das Senden eines anderen Logtyps an dieselbe Protokollgruppe einzurichten.

Protokollgruppe und Ressourcenrichtlinie

Die Protokollgruppe, an die die Protokolle gesendet werden, muss über eine Ressourcenrichtlinie verfügen, die bestimmte Berechtigungen enthält. Wenn die Protokollgruppe derzeit keine Ressourcenrichtlinie hat und der Benutzer, der die Protokollierung einrichtet, über die `logs:DescribeLogGroups` Berechtigungen `logs:PutResourcePolicy``logs:DescribeResourcePolicies`, und für die Protokollgruppe verfügt, erstellt er AWS automatisch die folgende Richtlinie für diese Gruppe, wenn Sie beginnen, die Protokolle an Logs zu CloudWatch senden. Für neu erstellte Abonnements werden die Ressourcenrichtlinien auf Protokollgruppenebene konfiguriert und haben eine maximale Größe von 51.200 Byte. Wenn eine bestehende Ressourcenrichtlinie auf Kontoebene bereits Berechtigungen

über Platzhalter gewährt, würde keine separate Richtlinie auf Protokollgruppenebene erstellt werden. Um die Group-level Protokollressourcenrichtlinie für eine bestimmte Protokollgruppe zu überprüfen, verwenden Sie den `describe-resource-policies` Befehl, bei dem der `--resource-arn` Parameter auf den Protokollgruppen-ARN und der `--policy-scope` Parameter auf gesetzt ist `RESOURCE`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite20150319",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "delivery.logs.amazonaws.com"
        ]
      },
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group:log-
stream:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": [
            "0123456789"
          ]
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:logs:us-east-1:111122223333:*"
          ]
        }
      }
    }
  ]
}
```

```
}
```

Das Ressourcenrichtlinien-Limit der Protokollgruppe beträgt 51.200 Byte. Sobald dieses Limit erreicht ist, kann AWS keine neuen Berechtigungen mehr hinzufügen. Dies erfordert, dass Kunden die Richtlinie manuell ändern, um dem `delivery.logs.amazonaws.com` Service-Principal Berechtigungen für die `logs:PutLogEvents` Aktionen `logs:CreateLogStream` und zu gewähren. Kunden sollten ein Präfix für den Protokollgruppennamen mit Platzhaltern wie z. B. verwenden `/aws/vendedlogs/*` und diesen Protokollgruppennamen für die future Erstellung von Lieferungen verwenden.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite20150319",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "delivery.logs.amazonaws.com"
        ]
      },
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group/aws/vendedlogs/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": [
            "0123456789"
          ]
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:logs:us-east-1:111122223333:*"
          ]
        }
      }
    }
  ]
}
```

```
}  
  ]  
    }  
      }  
        ]
```

An Amazon S3 gesendete Protokolle

Wenn Sie festlegen, dass Protokolle an Amazon S3 gesendet werden, werden bei Bedarf die Ressourcenrichtlinien für den S3-Bucket AWS erstellt oder geändert, der die Protokolle empfängt.

Protokolle, die direkt in Amazon S3 veröffentlicht werden, werden in einen bestehenden Bucket veröffentlicht, den Sie angeben. Alle fünf Minuten werden ein oder mehrere Protokolldateien in dem angegebenen Bucket erstellt.

Wenn Sie Protokolle zum ersten Mal an einen Amazon-S3-Bucket übermitteln, zeichnet der Service, der Protokolle ausstellt, den Besitzer des Buckets auf, um sicherzustellen, dass die Protokolle nur an einen Bucket gesendet werden, der zu diesem Konto gehört. Um den Amazon-S3-Bucket-Eigentümer zu ändern, müssen Sie daher das Protokollabonnement im ursprünglichen Service neu erstellen oder aktualisieren.

Note

CloudFront verwendet ein anderes Berechtigungsmodell als die anderen Dienste, die verkaufte Protokolle an S3 senden. Weitere Informationen finden Sie unter [Für die Konfiguration der Standard-Protokollierung und den Zugriff auf Ihre Protokolldateien erforderliche Berechtigungen](#).

Wenn Sie denselben S3-Bucket für CloudFront Zugriffsprotokolle und eine andere Protokollquelle verwenden, gewährt die Aktivierung von ACL auf dem Bucket für CloudFront außerdem allen anderen Protokollquellen, die diesen Bucket verwenden, Berechtigungen.

Important

Wenn Sie Logs an einen Amazon S3 S3-Bucket senden und die Bucket-Richtlinie ein `NotAction NotPrincipal` OR-Element enthält, schlägt das automatische Hinzufügen von Log-Lieferberechtigungen zum Bucket und das Erstellen eines Protokollabonnements

fehl. Um ein Protokollabonnement erfolgreich zu erstellen, müssen Sie der Bucket-Richtlinie manuell die Berechtigungen für die Protokollzustellung hinzufügen und dann das Protokollabonnement erstellen. Weitere Informationen finden Sie in den Anweisungen in diesem Abschnitt.

Wenn der Bucket über eine serverseitige Verschlüsselung mit einem vom Kunden verwalteten AWS KMS Schlüssel verfügt, müssen Sie auch die Schlüsselrichtlinie für Ihren vom Kunden verwalteten Schlüssel hinzufügen. Weitere Informationen finden Sie unter [Amazon S3](#).

Wenn für den Ziel-Bucket ein SSE-KMS Bucket Key aktiviert ist, funktioniert die angehängte, vom Kunden verwaltete KMS-Schlüsselrichtlinie nicht mehr wie erwartet für alle Anfragen. Weitere Informationen finden Sie unter [Senkung der Kosten SSE-KMS mit Amazon S3 Bucket Keys](#).

Wenn Sie Vended Logs und S3-Verschlüsselung mit einem vom Kunden verwalteten AWS KMS Schlüssel verwenden, müssen Sie bei der Konfiguration des Buckets einen vollqualifizierten AWS KMS Schlüssel-ARN anstelle einer Schlüssel-ID verwenden. Weitere Informationen finden Sie unter [Put-Bucket-Verschlüsselung](#).

Benutzerberechtigungen

Um das erstmalige Senden einer dieser Protokolltypen an Amazon S3 einrichten zu können, müssen Sie bei einem Konto mit den folgenden Berechtigungen angemeldet sein.

- `logs:CreateLogDelivery`
- `S3:GetBucketPolicy`
- `S3:PutBucketPolicy`

Wenn einer dieser Protokolltypen bereits an einen Amazon-S3-Bucket gesendet wird, benötigen Sie zum Einrichten eines weiteren Protokolls desselben Typs an dieselbe Protokollgruppe nur die `logs:CreateLogDelivery`-Berechtigung.

S3-Bucket-Ressourcenrichtlinie

Der S3-Bucket, an den die Protokolle gesendet werden, muss über eine Ressourcenrichtlinie verfügen, die bestimmte Berechtigungen enthält. Wenn der Bucket derzeit keine Ressourcenrichtlinie hat und der Benutzer, der die Protokollierung einrichtet, die `S3:GetBucketPolicy`- und

S3:PutBucketPolicy-Berechtigungen für den Bucket hat, dann erstellt AWS automatisch die folgende Richtlinie dafür, wenn Sie mit dem Senden der Protokolle an Amazon S3 beginnen.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "AWSLogDeliveryWrite20150319",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryAclCheck",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": [
            "0123456789"
          ]
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:logs:us-east-1:111122223333:*"
          ]
        }
      }
    },
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/AWSLogs/account-ID/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control",
          "aws:SourceAccount": [
            "0123456789"
          ]
        }
      }
    }
  ]
}
```

```

    ],
    },
    "ArnLike": {
      "aws:SourceArn": [
        "arn:aws:logs:us-east-1:111122223333:*"
      ]
    }
  }
}

```

Geben Sie in der vorherigen Richtlinie für `aws:SourceAccount` die Liste der Konto-IDs an, für die Protokolle an diesen Bucket übermittelt werden. Geben Sie für `aws:SourceArn` die Liste der ARNs der Ressource, die die Protokolle generiert, im Format `arn:aws:logs:source-region:source-account-id:*` an.

Wenn der Bucket über eine Ressourcenrichtlinie verfügt, diese Richtlinie jedoch nicht die in der vorherigen Richtlinie gezeigte Anweisung enthält und der Benutzer, der die Protokollierung einrichtet, die `S3:GetBucketPolicy`- und `S3:PutBucketPolicy`-Berechtigungen für den Bucket hat, wird diese Anweisung an die Ressourcenrichtlinie des Bucket angehängt.

Note

In einigen Fällen werden möglicherweise `AccessDenied` Fehler angezeigt, AWS CloudTrail wenn die `s3:ListBucket` Genehmigung nicht erteilt wurde. `delivery.logs.amazonaws.com` Um diese Fehler in Ihren CloudTrail Protokollen zu vermeiden, müssen Sie die `s3:ListBucket` Erlaubnis erteilen `delivery.logs.amazonaws.com` und die angegebenen `Condition` Parameter zusammen mit den in der vorherigen Bucket-Richtlinie festgelegten `s3:GetBucketAcl` Berechtigungen angeben. Zur Vereinfachung können Sie `AWSLogDeliveryAclCheck` direkt auf `"Action": ["s3:GetBucketAcl", "s3:ListBucket"]` aktualisieren, anstatt eine neue Anweisung (Statement) zu erstellen.

Serverseitige Verschlüsselung im Amazon-S3-Bucket

Sie können die Daten in Ihrem Amazon S3 S3-Bucket schützen, indem Sie entweder die serverseitige Verschlüsselung mit S3-managed Amazon-Schlüsseln (SSE-S3) oder die serverseitige

Verschlüsselung mit einem in AWS Key Management Service (SSE-KMS) gespeicherten AWS KMS Schlüssel aktivieren. Weitere Informationen finden Sie unter [Schützen von Daten mithilfe serverseitiger Verschlüsselung](#).

Wenn Sie möchten SSE-S3, ist keine zusätzliche Konfiguration erforderlich. Amazon S3 verarbeitet den Verschlüsselungsschlüssel.

 Warning

Wenn Sie möchten SSE-KMS, müssen Sie einen vom Kunden verwalteten Schlüssel verwenden, da die Verwendung eines AWS verwalteten Schlüssels in diesem Szenario nicht unterstützt wird. Wenn Sie die Verschlüsselung mit einem AWS verwalteten Schlüssel einrichten, werden die Protokolle in einem unlesbaren Format geliefert.

Wenn Sie einen vom Kunden verwalteten AWS KMS Schlüssel verwenden, können Sie den Amazon-Ressourcennamen (ARN) des vom Kunden verwalteten Schlüssels angeben, wenn Sie die Bucket-Verschlüsselung aktivieren. Sie müssen der Schlüsselrichtlinie für Ihren vom Kunden verwalteten Schlüssel (nicht der Bucket-Richtlinie für Ihren S3 Bucket) Folgendes hinzufügen, damit das Protokollzustellungskonto in Ihren S3 Bucket schreiben kann.

Wenn Sie möchten SSE-KMS, müssen Sie einen vom Kunden verwalteten Schlüssel verwenden, da die Verwendung eines AWS verwalteten Schlüssels in diesem Szenario nicht unterstützt wird. Wenn Sie einen vom Kunden verwalteten AWS KMS Schlüssel verwenden, können Sie den Amazon-Ressourcennamen (ARN) des vom Kunden verwalteten Schlüssels angeben, wenn Sie die Bucket-Verschlüsselung aktivieren. Sie müssen der Schlüsselrichtlinie für Ihren vom Kunden verwalteten Schlüssel (nicht der Bucket-Richtlinie für Ihren S3 Bucket) Folgendes hinzufügen, damit das Protokollzustellungskonto in Ihren S3 Bucket schreiben kann.

```
{
  "Sid": "Allow Logs Delivery to use the key",
  "Effect": "Allow",
  "Principal": {
    "Service": [ "delivery.logs.amazonaws.com" ]
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
```

```
    "kms:DescribeKey"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:SourceAccount": ["0123456789"]
    },
    "ArnLike": {
      "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:*"]
    }
  }
}
```

Geben Sie für `aws:SourceAccount` die Liste der Konto-IDs an, für die Protokolle an diesen Bucket übermittelt werden. Geben Sie für `aws:SourceArn` die Liste der ARNs der Ressource, die die Protokolle generiert, im Format `arn:aws:logs:source-region:source-account-id:*` an.

An Firehose gesendete Logs

Dieser Abschnitt gilt, wenn die in der Tabelle im vorherigen Abschnitt aufgeführten Protokolltypen an Firehose gesendet werden:

Benutzerberechtigungen

Um das erstmalige Senden dieser Arten von Protokollen an Firehose einrichten zu können, müssen Sie bei einem Konto mit den folgenden Berechtigungen angemeldet sein.

- `logs:CreateLogDelivery`
- `firehose:TagDeliveryStream`
- `iam:CreateServiceLinkedRole`

Wenn einer dieser Protokolltypen bereits an Firehose gesendet wurde, benötigen Sie nur die Berechtigungen `logs:CreateLogDelivery` und `firehose:TagDeliveryStream`, um das Senden eines anderen dieser Protokolltypen an Firehose einzurichten.

IAM-Rollen, die für Berechtigungen verwendet werden

Da Firehose keine Ressourcenrichtlinien AWS verwendet, verwendet es IAM-Rollen bei der Einrichtung dieser Protokolle, die an Firehose gesendet werden sollen. AWS erstellt eine

serviceverknüpfte Rolle mit dem Namen. `AWSServiceRoleForLogDelivery` Diese serviceverknüpfte Rolle umfasst die folgenden Berechtigungen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "firehose:PutRecord",
        "firehose:PutRecordBatch",
        "firehose:ListTagsForDeliveryStream"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/LogDeliveryEnabled": "true"
        }
      },
      "Effect": "Allow"
    }
  ]
}
```

Diese dienstbezogene Rolle gewährt Berechtigungen für alle Firehose-Lieferdatenströme, für die das `LogDeliveryEnabled` Tag auf gesetzt ist. `true` AWS weist dieses Tag dem Ziel-Lieferstream zu, wenn Sie die Protokollierung einrichten.

Diese serviceverknüpfte Rolle verfügt auch über eine Vertrauensrichtlinie, die es dem `delivery.logs.amazonaws.com`-Service-Prinzipal erlaubt, die erforderliche serviceverknüpfte Rolle zu übernehmen. Diese Vertrauensrichtlinie lautet wie folgt:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
    "Principal": {
      "Service": "delivery.logs.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
  }
]
```

Protokollierung, für die zusätzliche Berechtigungen [V2] erforderlich sind

Einige AWS Dienste verwenden eine neue Methode, um ihre Protokolle zu senden. Dies ist eine flexible Methode, mit der Sie die Protokollzustellung von diesen Services an eines oder mehrere der folgenden Ziele einrichten können: CloudWatch Logs, Amazon S3 oder Firehose und X-Ray für die Trace-Zustellung.

Eine funktionierende Protokollzustellung besteht aus drei Elementen:

- `ADeliverySource`, ein logisches Objekt, das die Ressource (n) darstellt, die die Protokolle tatsächlich senden.
- `ADeliveryDestination`, ein logisches Objekt, das das tatsächliche Lieferziel darstellt.
- `ADelivery`, das eine Lieferquelle mit einem Lieferziel verbindet

Um die Protokollzustellung zwischen einem unterstützten AWS Dienst und einem Ziel zu konfigurieren, müssen Sie wie folgt vorgehen:

- Erstellen Sie eine Lieferquelle mit [PutDeliverySource](#).
- Erstellen Sie ein Lieferziel mit [PutDeliveryDestination](#).
- Wenn Sie Protokolle kontoübergreifend versenden, müssen Sie dies [PutDeliveryDestinationPolicy](#) im Zielkonto verwenden, um dem Ziel eine IAM Richtlinie zuzuweisen. Diese Richtlinie autorisiert die Erstellung einer Lieferung von der Versandquelle in Konto A zum Lieferziel in Konto B. Für die kontoübergreifende Zustellung müssen Sie die Berechtigungsrichtlinien manuell selbst erstellen.
- Erstellen Sie eine Lieferung, indem Sie genau eine Zustellungsquelle und ein Lieferziel verknüpfen, indem Sie [CreateDelivery](#)

In den folgenden Abschnitten finden Sie Einzelheiten zu den Berechtigungen, die Sie benötigen, wenn Sie angemeldet sind, um die Protokollbereitstellung an die einzelnen Zieltypen mithilfe des V2-Prozesses einzurichten. Diese Berechtigungen können einer IAM-Rolle erteilt werden, mit der Sie angemeldet sind.

Important

Es liegt in Ihrer Verantwortung, Ressourcen für die Protokollzustellung zu entfernen, nachdem Sie die Ressource gelöscht haben, die das Protokoll generiert hat. Gehen Sie dazu wie folgt vor.

1. Löschen Sie das Delivery mithilfe der [DeleteDelivery](#) Operation.
2. Löschen Sie die DeliverySource mithilfe der [DeleteDeliverySource](#) Operation.
3. Wenn das mit dem DeliverySource, was Sie gerade gelöscht haben, DeliveryDestination verknüpft ist, nur für dieses spezielle DeliverySource Objekt verwendet wird, können Sie es mithilfe des [DeleteDeliveryDestinations](#) Vorgangs entfernen.

Inhalt

- [Protokolle, die an CloudWatch Logs gesendet wurden](#)
- [An Amazon S3 gesendete Protokolle](#)
 - [Serverseitige Verschlüsselung im Amazon-S3-Bucket](#)
- [An Firehose gesendete Logs](#)
- [Spuren wurden gesendet an X-Ray](#)

Protokolle, die an CloudWatch Logs gesendet wurden

Benutzerberechtigungen

Um das Senden von Protokollen an CloudWatch Logs zu ermöglichen, müssen Sie mit den folgenden Berechtigungen angemeldet sein.

JSON

```
{  
  "Version": "2012-10-17",
```



```

"Statement": [
  {
    "Sid": "ReadWriteAccessForLogDeliveryActions",
    "Effect": "Allow",
    "Action": [
      "logs:GetDelivery",
      "logs:GetDeliverySource",
      "logs:PutDeliveryDestination",
      "logs:GetDeliveryDestinationPolicy",
      "logs:DeleteDeliverySource",
      "logs:PutDeliveryDestinationPolicy",
      "logs:CreateDelivery",
      "logs:GetDeliveryDestination",
      "logs:PutDeliverySource",
      "logs:DeleteDeliveryDestination",
      "logs:DeleteDeliveryDestinationPolicy",
      "logs:DeleteDelivery",
      "logs:UpdateDeliveryConfiguration"
    ],
    "Resource": [
      "arn:aws:logs:us-east-1:111122223333:delivery:*",
      "arn:aws:logs:us-east-1:444455556666:delivery-source:*",
      "arn:aws:logs:us-east-1:777788889999:delivery-destination:*"
    ]
  },
  {
    "Sid": "ListAccessForLogDeliveryActions",
    "Effect": "Allow",
    "Action": [
      "logs:DescribeDeliveryDestinations",
      "logs:DescribeDeliverySources",
      "logs:DescribeDeliveries",
      "logs:DescribeConfigurationTemplates"
    ],
    "Resource": "*"
  },
  {
    "Sid": "AllowUpdatesToResourcePolicyCWL",
    "Effect": "Allow",
    "Action": [
      "logs:PutResourcePolicy",
      "logs:DescribeResourcePolicies",
      "logs:DescribeLogGroups"
    ]
  },

```

```

    "Resource": [
      "arn:aws:logs:us-east-1:123456789012:*"
    ]
  }
]
}

```

Protokollgruppe und Ressourcenrichtlinie

Die Protokollgruppe, an die die Protokolle gesendet werden, muss über eine Ressourcenrichtlinie verfügen, die bestimmte Berechtigungen enthält. Wenn die Protokollgruppe derzeit keine Ressourcenrichtlinie hat und der Benutzer, der die Protokollierung einrichtet, über die `logs:DescribeLogGroups` Berechtigungen `logs:PutResourcePolicy``logs:DescribeResourcePolicies`, und für die Protokollgruppe verfügt, erstellt er AWS automatisch die folgende Richtlinie für diese Gruppe, wenn Sie beginnen, die Protokolle an Logs zu CloudWatch senden. Für neu erstellte Abonnements werden die Ressourcenrichtlinien auf Protokollgruppenebene konfiguriert und haben eine maximale Größe von 51.200 Byte. Wenn eine bestehende Ressourcenrichtlinie auf Kontoebene bereits Berechtigungen über Platzhalter gewährt, würde keine separate Richtlinie auf Protokollgruppenebene erstellt werden. Um die Group-level Protokollressourcenrichtlinie für eine bestimmte Protokollgruppe zu überprüfen, verwenden Sie den `describe-resource-policies` Befehl, bei dem der `--resource-arn` Parameter auf den Protokollgruppen-ARN und der `--policy-scope` Parameter auf `gesetzt` ist `RESOURCE`.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite20150319",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "delivery.logs.amazonaws.com"
        ]
      },
      "Action": [
        "logs:CreateLogStream",

```

```

        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group:log-
stream:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": [
            "0123456789"
          ]
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:logs:us-east-1:111122223333:*"
          ]
        }
      }
    }
  ]
}

```

Das Ressourcenrichtlinien-Limit der Protokollgruppe beträgt 51.200 Byte. Sobald dieses Limit erreicht ist, kann AWS keine neuen Berechtigungen mehr hinzufügen. Dies erfordert, dass Kunden die Richtlinie manuell ändern, um dem `delivery.logs.amazonaws.com` Service-Principal Berechtigungen für die `logs:PutLogEvents` Aktionen `logs:CreateLogStream` und zu gewähren. Kunden sollten ein Präfix für den Protokollgruppennamen mit Platzhaltern wie z. B. verwenden `/aws/vendedlogs/*` und diesen Protokollgruppennamen für die future Erstellung von Lieferungen verwenden.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite20150319",
      "Effect": "Allow",
      "Principal": {
        "Service": [

```

```

        "delivery.logs.amazonaws.com"
    ],
    },
    "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": [
        "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group/aws/
vendedLogs/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": [
                "0123456789"
            ]
        },
        "ArnLike": {
            "aws:SourceArn": [
                "arn:aws:logs:us-east-1:111122223333:*"
            ]
        }
    }
}

```

An Amazon S3 gesendete Protokolle

Benutzerberechtigungen

Um das Senden von Protokollen an Amazon S3 zu aktivieren, müssen Sie mit den folgenden Berechtigungen angemeldet sein.

JSON

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccessForLogDeliveryActions",

```

```

      "Effect": "Allow",
      "Action": [
        "logs:GetDelivery",
        "logs:GetDeliverySource",
        "logs:PutDeliveryDestination",
        "logs:GetDeliveryDestinationPolicy",
        "logs>DeleteDeliverySource",
        "logs:PutDeliveryDestinationPolicy",
        "logs>CreateDelivery",
        "logs:GetDeliveryDestination",
        "logs:PutDeliverySource",
        "logs>DeleteDeliveryDestination",
        "logs>DeleteDeliveryDestinationPolicy",
        "logs>DeleteDelivery",
        "logs:UpdateDeliveryConfiguration"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:111122223333:delivery:*",
        "arn:aws:logs:us-east-1:111122223333:delivery-source:*",
        "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
      ]
    },
    {
      "Sid": "ListAccessForLogDeliveryActions",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeDeliveryDestinations",
        "logs:DescribeDeliverySources",
        "logs:DescribeDeliveries",
        "logs:DescribeConfigurationTemplates"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowUpdatesToResourcePolicyS3",
      "Effect": "Allow",
      "Action": [
        "s3:PutBucketPolicy",
        "s3:GetBucketPolicy"
      ],
      "Resource": "arn:aws:s3:::bucket-name"
    }
  ]

```

}

Der S3-Bucket, an den die Protokolle gesendet werden, muss über eine Ressourcenrichtlinie verfügen, die bestimmte Berechtigungen enthält. Wenn der Bucket derzeit keine Ressourcenrichtlinie hat und der Benutzer, der die Protokollierung einrichtet, die `S3:GetBucketPolicy`- und `S3:PutBucketPolicy`-Berechtigungen für den Bucket hat, dann erstellt AWS automatisch die folgende Richtlinie dafür, wenn Sie mit dem Senden der Protokolle an Amazon S3 beginnen.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "AWSLogDeliveryWrite20150319",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/AWSLogs/account-ID/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control",
          "aws:SourceAccount": [
            "0123456789"
          ]
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:logs:us-east-1:111122223333:delivery-source:*"
          ]
        }
      }
    }
  ]
}
```

Geben Sie in der vorherigen Richtlinie für `aws:SourceAccount` die Liste der Konto-IDs an, für die Protokolle an diesen Bucket übermittelt werden. Geben Sie für `aws:SourceArn` die Liste der ARNs der Ressource, die die Protokolle generiert, im Format `arn:aws:logs:source-region:source-account-id:*` an.

Wenn der Bucket über eine Ressourcenrichtlinie verfügt, diese Richtlinie jedoch nicht die in der vorherigen Richtlinie gezeigte Anweisung enthält und der Benutzer, der die Protokollierung einrichtet, die `S3:GetBucketPolicy`- und `S3:PutBucketPolicy`-Berechtigungen für den Bucket hat, wird diese Anweisung an die Ressourcenrichtlinie des Bucket angehängt.

Note

In einigen Fällen werden möglicherweise `AccessDenied` Fehler angezeigt, AWS CloudTrail wenn die `s3:ListBucket` Genehmigung nicht erteilt wurde. `delivery.logs.amazonaws.com` Um diese Fehler in Ihren CloudTrail Protokollen zu vermeiden, müssen Sie die `s3:ListBucket` Erlaubnis erteilen `delivery.logs.amazonaws.com` und die angegebenen `Condition` Parameter zusammen mit den in der vorherigen Bucket-Richtlinie festgelegten `s3:GetBucketAcl` Berechtigungen angeben. Zur Vereinfachung können Sie `AWSLogDeliveryAclCheck` direkt auf `"Action": ["s3:GetBucketAcl", "s3:ListBucket"]` aktualisieren, anstatt eine neue Anweisung (Statement) zu erstellen.

Serverseitige Verschlüsselung im Amazon-S3-Bucket

Sie können die Daten in Ihrem Amazon S3 S3-Bucket schützen, indem Sie entweder die serverseitige Verschlüsselung mit S3-managed Amazon-Schlüsseln (SSE-S3) oder die serverseitige Verschlüsselung mit einem in AWS Key Management Service (SSE-KMS) gespeicherten AWS KMS Schlüssel aktivieren. Weitere Informationen finden Sie unter [Schützen von Daten mithilfe serverseitiger Verschlüsselung](#).

Wenn Sie möchten SSE-S3, ist keine zusätzliche Konfiguration erforderlich. Amazon S3 verarbeitet den Verschlüsselungsschlüssel.

Warning

Wenn Sie möchten SSE-KMS, müssen Sie einen vom Kunden verwalteten Schlüssel verwenden, da die Verwendung eines AWS verwalteten Schlüssels in diesem Szenario

nicht unterstützt wird. Wenn Sie die Verschlüsselung mit einem AWS verwalteten Schlüssel einrichten, werden die Protokolle in einem unlesbaren Format geliefert.

Wenn Sie einen vom Kunden verwalteten AWS KMS Schlüssel verwenden, können Sie den Amazon-Ressourcennamen (ARN) des vom Kunden verwalteten Schlüssels angeben, wenn Sie die Bucket-Verschlüsselung aktivieren. Sie müssen der Schlüsselrichtlinie für Ihren vom Kunden verwalteten Schlüssel (nicht der Bucket-Richtlinie für Ihren S3 Bucket) Folgendes hinzufügen, damit das Protokollzustellungskonto in Ihren S3 Bucket schreiben kann.

Wenn Sie möchten SSE-KMS, müssen Sie einen vom Kunden verwalteten Schlüssel verwenden, da die Verwendung eines AWS verwalteten Schlüssels in diesem Szenario nicht unterstützt wird. Wenn Sie einen vom Kunden verwalteten AWS KMS Schlüssel verwenden, können Sie den Amazon-Ressourcennamen (ARN) des vom Kunden verwalteten Schlüssels angeben, wenn Sie die Bucket-Verschlüsselung aktivieren. Sie müssen der Schlüsselrichtlinie für Ihren vom Kunden verwalteten Schlüssel (nicht der Bucket-Richtlinie für Ihren S3 Bucket) Folgendes hinzufügen, damit das Protokollzustellungskonto in Ihren S3 Bucket schreiben kann.

```
{
  "Sid": "Allow Logs Delivery to use the key",
  "Effect": "Allow",
  "Principal": {
    "Service": [ "delivery.logs.amazonaws.com" ]
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:SourceAccount": ["0123456789"]
    },
    "ArnLike": {
      "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:delivery-source:*"]
    }
  }
}
```



```
}
```

Geben Sie für `aws:SourceAccount` die Liste der Konto-IDs an, für die Protokolle an diesen Bucket übermittelt werden. Geben Sie für `aws:SourceArn` die Liste der ARNs der Ressource, die die Protokolle generiert, im Format `arn:aws:logs:source-region:source-account-id:*` an.

An Firehose gesendete Logs

Benutzerberechtigungen

Um das Senden von Protokollen an Firehose zu ermöglichen, müssen Sie mit den folgenden Berechtigungen angemeldet sein.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadWriteAccessForLogDeliveryActions",
      "Effect": "Allow",
      "Action": [
        "logs:GetDelivery",
        "logs:GetDeliverySource",
        "logs:PutDeliveryDestination",
        "logs:GetDeliveryDestinationPolicy",
        "logs>DeleteDeliverySource",
        "logs:PutDeliveryDestinationPolicy",
        "logs>CreateDelivery",
        "logs:GetDeliveryDestination",
        "logs:PutDeliverySource",
        "logs>DeleteDeliveryDestination",
        "logs>DeleteDeliveryDestinationPolicy",
        "logs>DeleteDelivery",
        "logs:UpdateDeliveryConfiguration"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:111122223333:delivery:*",
        "arn:aws:logs:us-east-1:111122223333:delivery-source:*",
        "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
      ]
    }
  ]
}
```

```

    },
    {
      "Sid": "ListAccessForLogDeliveryActions",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeDeliveryDestinations",
        "logs:DescribeDeliverySources",
        "logs:DescribeDeliveries",
        "logs:DescribeConfigurationTemplates"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowUpdatesToResourcePolicyFH",
      "Effect": "Allow",
      "Action": [
        "firehose:TagDeliveryStream"
      ],
      "Resource": [
        "arn:aws:firehose:us-east-1:111122223333:deliverystream/*"
      ]
    },
    {
      "Sid": "CreateServiceLinkedRole",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": "arn:aws:iam::111122223333:role/aws-service-role/delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery"
    }
  ]
}

```

IAM-Rollen, die für Ressourcenberechtigungen verwendet werden

Da Firehose keine Ressourcenrichtlinien AWS verwendet, verwendet es IAM-Rollen bei der Einrichtung dieser Protokolle, die an Firehose gesendet werden sollen. AWS erstellt eine serviceverknüpfte Rolle mit dem Namen. `AWSServiceRoleForLogDelivery` Diese serviceverknüpfte Rolle umfasst die folgenden Berechtigungen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "firehose:PutRecord",
        "firehose:PutRecordBatch",
        "firehose:ListTagsForDeliveryStream"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/LogDeliveryEnabled": "true"
        }
      },
      "Effect": "Allow"
    }
  ]
}
```

Diese dienstbezogene Rolle gewährt Berechtigungen für alle Firehose-Lieferdatenströme, für die das `LogDeliveryEnabled` Tag auf gesetzt ist. `true` AWS weist dieses Tag dem Ziel-Lieferstream zu, wenn Sie die Protokollierung einrichten.

Diese serviceverknüpfte Rolle verfügt auch über eine Vertrauensrichtlinie, die es dem `delivery.logs.amazonaws.com`-Service-Prinzipal erlaubt, die erforderliche serviceverknüpfte Rolle zu übernehmen. Diese Vertrauensrichtlinie lautet wie folgt:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },

```

```
    "Action": "sts:AssumeRole"
  }
]
}
```

Spuren wurden gesendet an X-Ray

Benutzerberechtigungen

Um das Senden von Traces an zu aktivieren AWS X-Ray, müssen Sie mit den folgenden Berechtigungen angemeldet sein.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadWriteAccessForLogDeliveryActions",
      "Effect": "Allow",
      "Action": [
        "logs:GetDelivery",
        "logs:GetDeliverySource",
        "logs:PutDeliveryDestination",
        "logs:GetDeliveryDestinationPolicy",
        "logs>DeleteDeliverySource",
        "logs:PutDeliveryDestinationPolicy",
        "logs>CreateDelivery",
        "logs:GetDeliveryDestination",
        "logs:PutDeliverySource",
        "logs>DeleteDeliveryDestination",
        "logs>DeleteDeliveryDestinationPolicy",
        "logs>DeleteDelivery",
        "logs:UpdateDeliveryConfiguration"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:111122223333:delivery:*",
        "arn:aws:logs:us-east-1:111122223333:delivery-source:*",
        "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
      ]
    }
  ],
}
```

```

    {
      "Sid": "ListAccessForLogDeliveryActions",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeDeliveryDestinations",
        "logs:DescribeDeliverySources",
        "logs:DescribeDeliveries",
        "logs:DescribeConfigurationTemplates"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowUpdatesToResourcePolicyXRay",
      "Effect": "Allow",
      "Action": [
        "xray:PutResourcePolicy",
        "xray:ListResourcePolicies",
        "xray:GetTraceSegmentDestination"
      ],
      "Resource": "*"
    }
  ]
}

```

X-Ray Ressourcenrichtlinie

Das Zielkonto, an das die Traces gesendet werden, muss über eine Ressourcenrichtlinie verfügen, die bestimmte Berechtigungen beinhaltet. Wenn der Benutzer, der die Ablaufverfolgung einrichtet, über `xray:ListResourcePolicies` Berechtigungen im Konto verfügt, `xray:PutResourcePolicy`, AWS wird die Ressourcenrichtlinie automatisch erstellt, wenn Sie mit dem Senden von Ablaufverfolgungen beginnen. X-Ray Welche Richtlinie erstellt wird, hängt vom Quelldienst ab:

Amazon Bedrock AgentCore Ressourcen

AWS erstellt eine Ressourcenrichtlinie pro Ressourcentyp. Die Richtlinie verwendet Platzhaltermuster, die bis zur Kontogrenze reichen und alle Ressourcen desselben Amazon Bedrock AgentCore Ressourcentyps im Konto abdecken. Wenn eine Amazon Bedrock AgentCore Speicherressource beispielsweise für die Ablaufverfolgung aktiviert ist, deckt die Richtlinie alle Speicherressourcen in diesem Konto ab — einschließlich aller Speicherressourcen, die in future erstellt werden.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "xray:PutTraceSegments",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ForAllValues:ArnLike": {
          "logs:LogGeneratingResourceArns": "arn:aws:bedrock-agentcore:us-east-1:123456789012:memory/*"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:logs:us-east-1:123456789012:delivery-source:*"
        }
      }
    }
  ]
}
```

Andere AWS Dienste

AWS Erstellt für andere Dienste, die die Ablaufverfolgung unterstützen, eine Ressourcenrichtlinie, die auf die jeweilige Quellressource beschränkt ist.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
```

```

    "Principal": {
      "Service": "delivery.logs.amazonaws.com"
    },
    "Action": "xray:PutTraceSegments",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ForAllValues:ArnLike": {
        "logs:LogGeneratingResourceArns": "arn:aws:bedrock:us-east-1:123456789012:knowledge-base/KnowledgeBaseId"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:logs:us-east-1:123456789012:delivery-source:xray-test"
      }
    }
  }
]
}

```

Aktivieren Sie die Transaktionssuche

Um das Senden von Traces an zu aktivieren X-Ray, müssen Sie die [Transaktionssuche](#) aktivieren.

Service-specific Berechtigungen

Zusätzlich zu den zielspezifischen Berechtigungen, die in den vorherigen Abschnitten aufgeführt wurden, erfordern einige Dienste als zusätzliche Sicherheitsebene die ausdrückliche Autorisierung, dass Kunden Protokolle von ihren Ressourcen aus senden dürfen. Es autorisiert die `AllowVendedLogDeliveryForResource` Aktion für Ressourcen, die Protokolle innerhalb dieses Dienstes verkaufen. Verwenden Sie für diese Dienste die folgende Richtlinie *service* und *resource-type* ersetzen Sie sie durch die entsprechenden Werte. Die dienstspezifischen Werte für diese Felder finden Sie auf der Dokumentationsseite dieser Dienste für verkaufte Protokolle. Im folgenden Beispiel wurde die Richtlinie aktualisiert, um verkaufte Protokolle von Amazon SES zu ermöglichen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ServiceLevelAccessForLogDelivery",
      "Effect": "Allow",
      "Action": [
        "ses:AllowVendedLogDeliveryForResource"
      ],
      "Resource": "arn:aws:ses:us-east-1:123456789012:resource-type/*"
    }
  ]
}
```

Console-specific Berechtigungen

Wenn Sie die Protokollzustellung über die Konsole statt über die APIs einrichten, benötigen Sie zusätzlich zu den in den vorherigen Abschnitten aufgeführten Berechtigungen auch die folgenden zusätzlichen Berechtigungen:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowLogDeliveryActionsConsoleCWL",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups",
        "logs:CreateLogGroup"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:111122223333:log-group:*"
      ]
    },
    {
      "Sid": "AllowLogDeliveryActionsConsoleS3",
```



```

        "Effect": "Allow",
        "Action": [
            "s3:ListAllMyBuckets",
            "s3:ListBucket",
            "s3:GetBucketLocation"
        ],
        "Resource": [
            "arn:aws:s3:::*"
        ]
    },
    {
        "Sid": "AllowLogDeliveryActionsConsoleFH",
        "Effect": "Allow",
        "Action": [
            "firehose:ListDeliveryStreams",
            "firehose:DescribeDeliveryStream"
        ],
        "Resource": [
            "*"
        ]
    }
]
}

```

Cross-account Beispiel für eine Zustellung

In diesem Beispiel sind zwei Konten betroffen. Das Konto mit der Ressource, die das Protokoll generiert, ist Konto A, ID: **123456789012**, und das Konto mit der Ressource, die das Protokoll verwendet, ist Konto B, ID: **111122223333**

Konto A möchte Logs aus der Amazon Bedrock Wissensdatenbank in seinem Konto mit dem ARN `arn:aws:bedrock: :knowledge-base/` bereitstellen. ***us-east-1 123456789012 kb-12345678***

Für dieses Beispiel benötigt Konto A die folgenden Berechtigungen:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {

```

```

    "Sid": "AllowVendedLogDeliveryForKnowledgeBase",
    "Effect": "Allow",
    "Action": [
        "bedrock:AllowVendedLogDeliveryForResource"
    ],
    "Resource": "arn:aws:bedrock:us-east-1:123456789012:knowledge-
base/XXXXXXXXXX"
  },
  {
    "Sid": "CreateLogDeliveryPermissions",
    "Effect": "Allow",
    "Action": [
        "logs:PutDeliverySource",
        "logs:CreateDelivery"
    ],
    "Resource": [
        "arn:aws:logs:us-east-1:123456789012:delivery-source:*",
        "arn:aws:logs:us-east-1:123456789012:delivery:*",
        "arn:aws:logs:us-east-1:444455556666:delivery-destination:*"
    ]
  }
]
}

```

Lieferquelle erstellen

Zunächst erstellt Konto A eine Lieferquelle mit ihrer grundlegenden Wissensdatenbank:

```
aws logs put-delivery-source --name my-delivery-source --log-type APPLICATION_LOGS --
resource-arn arn:aws:bedrock:region:AAAAAAAAAAAA:knowledge-base/XXXXXXXXXX
```

Als Nächstes muss Konto B das Lieferziel mithilfe eines der folgenden Verfahren erstellen:

- [Versand an einen Amazon S3 S3-Bucket konfigurieren](#)
- [Konfigurieren Sie die Lieferung an einen Firehose-Stream](#)

Versand an einen Amazon S3 S3-Bucket konfigurieren

Konto B möchte die Logs in seinem S3-Bucket mit dem ARN `arn:aws:s3:::amzn-s3-demo-bucket` empfangen. In diesem Beispiel benötigt Konto B die folgenden Berechtigungen:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PutLogDestinationPermissions",
      "Effect": "Allow",
      "Action": [
        "logs:PutDeliveryDestination",
        "logs:PutDeliveryDestinationPolicy"
      ],
      "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
    }
  ]
}
```

Der Bucket benötigt die folgenden Berechtigungen in seiner Bucket-Richtlinie:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogsDeliveryWrite",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/AWSLogs/123456789012/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control",
          "aws:SourceAccount": [
            "123456789012"
          ]
        }
      }
    }
  ]
}
```

```

    ]
  },
  "ArnLike": {
    "aws:SourceArn": [
      "arn:aws:logs:us-east-1:123456789012:delivery-source:my-
delivery-source"
    ]
  }
}
]
}

```

Wenn der Bucket mit verschlüsselt ist SSE-KMS, stellen Sie sicher, dass die AWS KMS Schlüsselrichtlinie über die entsprechenden Berechtigungen verfügt. Wenn der KMS-Schlüssel beispielsweise lautet `arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`, verwenden Sie Folgendes:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowLogsGenerateDataKey",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKey"
      ],
      "Resource": "arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": [
            "123456789012"
          ]
        }
      },
    },
  ],
}

```

```

        "ArnLike": {
            "aws:SourceArn": [
                "arn:aws:logs:us-east-1:123456789012:delivery-source:my-
delivery-source"
            ]
        }
    }
}

```

Konto B kann dann ein Lieferziel mit dem S3-Bucket als Zielressource erstellen:

```
aws logs put-delivery-destination --name my-s3-delivery-destination --delivery-
destination-configuration "destinationResourceArn=arn:aws:s3:::amzn-s3-demo-bucket"
```

Als Nächstes erstellt Konto B eine Lieferzielrichtlinie für das neu erstellte Lieferziel, die Konto A die Erlaubnis erteilt, eine Protokollzustellung zu erstellen. Die Richtlinie, die dem neu erstellten Lieferziel hinzugefügt wird, lautet wie folgt:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateDelivery",
      "Effect": "Allow",
      "Principal": {
        "AWS": "123456789012"
      },
      "Action": [
        "logs:CreateDelivery"
      ],
      "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:amzn-s3-demo-bucket"
    }
  ]
}

```

Diese Richtlinie wird auf dem Computer von Konto B gespeichert. `destination-policy-s3.json`
Um diese Ressource anzuhängen, führt Konto B den folgenden Befehl aus:

```
aws logs put-delivery-destination-policy --delivery-destination-name my-s3-delivery-destination --delivery-destination-policy file://destination-policy-s3.json
```

Schließlich erstellt Konto A die Lieferung, wodurch die Versandquelle in Konto A mit dem Lieferziel in Konto B verknüpft wird.

```
aws logs create-delivery --delivery-source-name my-delivery-source --delivery-destination-arn arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:my-s3-delivery-destination
```

Konfigurieren Sie die Lieferung an einen Firehose-Stream

In diesem Beispiel möchte Konto B Logs in seinen Firehose-Stream empfangen. Der Firehose-Stream hat den folgenden ARN und ist für die Verwendung des DirectPut Delivery-Stream-Typs konfiguriert:

`arn:aws:firehose:us-east-1:111122223333:deliverystream/log-delivery-stream`

Für dieses Beispiel benötigt Konto B die folgenden Berechtigungen:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFirehoseCreateSLR",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": "arn:aws:iam::111122223333:role/aws-service-role/delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery"
    },
    {
      "Sid": "AllowFirehoseTagging",
      "Effect": "Allow",
      "Action": [
```

```

        "firehose:TagDeliveryStream"
      ],
      "Resource": "arn:aws:firehose:us-
east-1:111122223333:deliverystream/X"
    },
    {
      "Sid": "AllowFirehoseDeliveryDestination",
      "Effect": "Allow",
      "Action": [
        "logs:PutDeliveryDestination",
        "logs:PutDeliveryDestinationPolicy"
      ],
      "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:*"
    }
  ]
}

```

Für den Firehose-Stream muss das Tag auf `LogDeliveryEnabled` gesetzt sein `true`.

Konto B erstellt dann ein Lieferziel mit dem Firehose-Stream als Zielressource:

```

aws logs put-delivery-destination --name my-fh-delivery-destination --delivery-
destination-configuration
  "destinationResourceArn=arn:aws:firehose:region:BBBBBBBBBBBBBB:deliverystream/X"

```

Als Nächstes erstellt Konto B eine Lieferzielrichtlinie für das neu erstellte Lieferziel, die Konto A die Erlaubnis erteilt, eine Protokollzustellung zu erstellen. Die Richtlinie, die dem neu erstellten Lieferziel hinzugefügt werden soll, lautet wie folgt:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateDelivery",
      "Effect": "Allow",
      "Principal": {
        "AWS": "123456789012"
      },

```

```

    "Action": [
        "logs:CreateDelivery"
    ],
    "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:amzn-s3-demo-bucket"
    }
}

```

Diese Richtlinie wird auf dem Computer von Konto B gespeichert. `destination-policy-fh.json`
Um diese Ressource anzuhängen, führt Konto B den folgenden Befehl aus:

```
aws logs put-delivery-destination-policy --delivery-destination-name my-fh-delivery-
destination --delivery-destination-policy file://destination-policy-fh.json
```

Schließlich erstellt Konto A die Lieferung, wodurch die Versandquelle in Konto A mit dem Lieferziel in Konto B verknüpft wird.

```
aws logs create-delivery --delivery-source-name my-delivery-source --delivery-
destination-arn arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:my-fh-delivery-
destination
```

Cross-service verwirrter Stellvertreter, Prävention

Das Problem des verwirrten Stellvertreters ist ein Sicherheitsproblem, bei dem eine Entität, die keine Berechtigung zur Durchführung einer Aktion hat, eine privilegiertere Entität zur Durchführung der Aktion zwingen kann. Im AWS Fall eines dienststellenübergreifenden Identitätswechsels kann es zu einem Problem mit verwirrten Stellvertretern kommen. Cross-service Ein Identitätswechsel kann auftreten, wenn ein Dienst (der anrufende Dienst) einen anderen Dienst (den angerufenen Dienst) aufruft. Der Anruf-Dienst kann so manipuliert werden, dass er seine Berechtigungen verwendet, um auf die Ressourcen eines anderen Kunden zu reagieren, auf die er sonst nicht zugreifen dürfte. Um dies zu verhindern, werden Tools AWS bereitgestellt, mit denen Sie Ihre Daten für alle Dienste mit Dienstprinzipalen schützen können, denen Zugriff auf Ressourcen in Ihrem Konto gewährt wurde.

Wir empfehlen [aws:SourceArn](#), die Kontextschlüssel, [aws:SourceAccount](#)[aws:SourceOrgID](#), und die [aws:SourceOrgPaths](#) globalen Bedingungsschlüssel in Ressourcenrichtlinien zu verwenden, um die Berechtigungen einzuschränken, die CloudWatch Logs einem anderen Dienst für die Ressource gewährt. Verwenden Sie `aws:SourceArn`, um nur einer Ressource

den serviceübergreifenden Zugriff zuzuordnen. Verwenden Sie `aws:SourceAccount`, um jede Ressource in diesem Konto der serviceübergreifenden Nutzung zuzuordnen. Verwenden Sie `aws:SourceOrgID`, damit alle Ressourcen von allen Konten innerhalb einer Organisation der serviceübergreifenden Nutzung zugeordnet werden können. Wird verwendet `aws:SourceOrgPaths`, um jede Ressource aus Konten innerhalb eines AWS Organizations Pfads der dienstübergreifenden Nutzung zuzuordnen. Weitere Informationen zur Verwendung und zum Verständnis von Pfaden finden Sie unter [Grundlegendes zum AWS Organizations Entitätspfad](#).

Der effektivste Weg, um sich vor dem Confused-Deputy-Problem zu schützen, ist die Verwendung des globalen Bedingungskontext-Schlüssels `aws:SourceArn` mit dem vollständigen ARN der Ressource. Wenn Sie den vollständigen ARN der Ressource nicht kennen oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den globalen Kontextbedingungsschlüssel `aws:SourceArn` mit Platzhalterzeichen (*) für die unbekannten Teile des ARN. Beispiel, `arn:aws:servicename:*:123456789012:*`.

Wenn der `aws:SourceArn`-Wert die Konto-ID nicht enthält, z. B. einen Amazon-S3-Bucket-ARN, müssen Sie sowohl `aws:SourceAccount` als auch `aws:SourceArn` verwenden, um Berechtigungen einzuschränken.

Um sich vor dem Confused-Deputy-Problem im großen Maßstab zu schützen, verwenden Sie den globalen Bedingungskontextschlüssel `aws:SourceOrgID` oder `aws:SourceOrgPaths` mit der Organisations-ID oder dem Organisationspfad der Ressource in Ihren ressourcenbasierten Richtlinien. Richtlinien, die den Schlüssel `aws:SourceOrgID` oder `aws:SourceOrgPaths` enthalten, schließen automatisch die richtigen Konten ein und Sie müssen die Richtlinien nicht manuell aktualisieren, wenn Sie Konten in Ihrer Organisation hinzufügen, entfernen oder verschieben.

Die dokumentierten Richtlinien zur Gewährung des Zugriffs auf CloudWatch Logs zum Schreiben von Daten in Kinesis Data Streams und Firehose in [Schritt 1: Erstellen eines Ziels](#) und [Schritt 2: Erstellen eines Ziels](#) zeigen, wie Sie den `aws:SourceArn` globalen Bedingungskontextschlüssel verwenden können, um das Problem des verwirrten Stellvertreters zu verhindern.

Automatisieren Sie die Aktivierung von Protokollen mit Regeln zur Telemetrieaktivierung

Sie können Regeln zur Aktivierung der Telemetrie verwenden, um die Protokollerfassung für Ihre Ressourcen automatisch zu konfigurieren. AWS Regeln helfen Ihnen dabei, die Protokollerfassung in Ihrer Organisation oder Ihren Konten zu standardisieren und eine konsistente Überwachung sicherzustellen.

Mit Aktivierungsregeln können Sie:

- Aktivieren Sie automatisch die Protokollierung für neue und bestehende Ressourcen, die Ihrem Regelbereich entsprechen
- Wenden Sie Regeln auf Organisations-, Organisationseinheit- (OU) oder Einzelkontoebene an
- Filtern Sie mithilfe von Tags, auf welche Ressourcen sich eine Regel auswirkt

Aktivierungsregeln unterstützen derzeit die folgenden AWS Telemetriequellen: Amazon VPC Flow Logs, Logs, Route 53 Resolver Query AWS WAF Logs, NLB Access Logs, Amazon EKS Control Plane Logs, CloudTrail Daten- und Verwaltungsereignisse, Amazon Bedrock AgentCore Logs, Amazon EC2 Detailed Metrics, AWS Security Hub, Amazon Bedrock Gateway, Amazon Bedrock AgentCore Memory und Distribution. AgentCore CloudFront

Vollständige Informationen zu Aktivierungsregeln, einschließlich Regelverhalten, Regelverwaltung, Fehlerbehebung und servicespezifischen Überlegungen, finden Sie unter [Telemetrieaktivierungsregeln](#) im Amazon-Benutzerhandbuch. CloudWatch

Hierarchie der Regel-Evaluierung

Aktivierungsregeln werden hierarchisch bewertet: zuerst Organisationsregeln, dann Regeln, dann OU-level Regeln auf Kontoebene. Regeln auf höheren Ebenen stellen die grundlegende Telemetrie bereit. Lower-level Regeln können zusätzliche Telemetriedaten hinzufügen, sie aber nicht reduzieren. Wenn widersprüchliche Regeln für denselben Geltungsbereich existieren, werden keine angewendet, bis der Konflikt gelöst ist.

Eine Aktivierungsregel erstellen

Um eine Regel zur Aktivierung von Telemetrie zu erstellen

1. Öffnen Sie die CloudWatch Konsole unter. <https://console.aws.amazon.com/cloudwatch/>
2. Wählen Sie im Navigationsbereich Telemetriedatenkonfiguration aus.
3. Wählen Sie die Registerkarte Aktivierungsregeln aus.
4. Wählen Sie Regel hinzufügen aus.
5. Geben Sie den Regelnamen, den Geltungsbereich (Organisation, Organisationseinheit oder Konto), die Datenquelle und den Telemetriesyp an.
6. Fügen Sie optional Tags hinzu, um zu filtern, auf welche Ressourcen sich die Regel auswirkt.
7. Wählen Sie Regel erstellen aus.

Aktivieren Sie die Protokollierung aus Quellen von Drittanbietern

CloudWatch Logs unterstützt die Aufnahme von Protokollen aus Drittanbieterquellen über direkte API-Integrationen und Amazon S3 S3-Bucket-Integrationen. Sie können auch zusätzliche Sicherheitserkenntnisse von Drittanbietern über AWS Security Hub CSPM erhalten.

Direkte Integrationen von Drittanbietern

CloudWatch Logs bietet direkte Integrationen mit den folgenden Quellen von Drittanbietern. Diese Integrationen verwenden entweder direkte API-Verbindungen oder Amazon S3 S3-Bucket-Integrationen, um Logs in Logs aufzunehmen: CloudWatch

- CrowdStrike Falcon
- Microsoft Office 365
- Okta Auth0
- Microsoft Entra ID
- Palo Alto Networks NGFW
- Microsoft Windows-Ereignisse
- Wiz
- Internetzugang mit Zscaler
- Okta SSO
- SentinelOne Endpunktsicherheit
- GitHub Audit-Protokolle
- ServiceNow CMDB
- Cisco Regenschirm

Anweisungen zur Einrichtung finden Sie in der [Anleitung zur Einrichtung der Drittanbieter-Integration](#) im CloudWatch Amazon-Benutzerhandbuch.

Zusätzliche Quellen über AWS Security Hub CSPM

Zusätzlich zu den direkten Integrationen senden Sicherheitspartner von Drittanbietern Ergebnisse an AWS Security Hub CSPM, die dann als Datenquellen in Logs verfügbar sind. CloudWatch In der folgenden Tabelle sind die Security Hub CSPM-Partnerintegrationen und deren Integrationstyp aufgeführt.

Um Security Hub CSPM-Ergebnisse als Datenquelle in CloudWatch Logs zu aktivieren, erstellen Sie in der Konsole eine Telemetrieaktivierungsregel für AWS Security Hub. CloudWatch Die Aktivierungsregel ist so konfiguriert, CloudWatch dass Ergebnisse aus Security Hub CSPM automatisch in eine verwaltete Protokollgruppe aufgenommen werden. step-by-stepAnweisungen finden Sie unter [Regeln zur Aktivierung der Telemetrie](#) im CloudWatch Amazon-Benutzerhandbuch.

Partner	Integration
3 — NTA CORESec	Sendet Ergebnisse über Security Hub CSPM
Alert Logic — Bedrohungsmanagement SIEMless	Sendet Ergebnisse über Security Hub CSPM
Aqua Security — Cloud-native Sicherheitsplattform	Sendet Ergebnisse über Security Hub CSPM
Aqua Security — KUBE-Bench	Sendet Ergebnisse über Security Hub CSPM
Rüstung — Rüstung überall	Sendet Ergebnisse über Security Hub CSPM
AttackIQ	Sendet Ergebnisse über Security Hub CSPM
Barracuda Networks — Hüter der Cloud-Sicherheit	Sendet Ergebnisse über Security Hub CSPM
BigID — BigID Enterprise	Sendet Ergebnisse über Security Hub CSPM

Partner	Integration
Blaues Sechseck	Sendet Ergebnisse über Security Hub CSPM
Checkpoint — CloudGuard IaaS	Sendet Ergebnisse über Security Hub CSPM
Check Point — Posture Management CloudGuard	Sendet Ergebnisse über Security Hub CSPM
Clarity — Kuppel	Sendet Ergebnisse über Security Hub CSPM
Cloud-Speichersicherheit — Antivirus für Amazon S3	Sendet Ergebnisse über Security Hub CSPM
Contrast Security — Contrast Assess	Sendet Ergebnisse über Security Hub CSPM
CrowdStrike — Falke CrowdStrike	Sendet Ergebnisse über Security Hub CSPM
CyberArk — Analyse privilegierter Bedrohungen	Sendet Ergebnisse über Security Hub CSPM
Satz von Daten	Sendet Ergebnisse über Security Hub CSPM
Daten	Sendet Ergebnisse über Security Hub CSPM
Forcepoint — CASB	Sendet Ergebnisse über Security Hub CSPM
Forcepoint — Cloud-Sicherheitsgateway	Sendet Ergebnisse über Security Hub CSPM
Forcepoint — DLP	Sendet Ergebnisse über Security Hub CSPM

Partner	Integration
Forcepoint — NGFW	Sendet Ergebnisse über Security Hub CSPM
Fuge	Sendet Ergebnisse über Security Hub CSPM
Guardicore — Centra	Sendet Ergebnisse über Security Hub CSPM
HackerOne — Informationen zu Sicherheitslücken	Sendet Ergebnisse über Security Hub CSPM
JFrog — Röntgen	Sendet Ergebnisse über Security Hub CSPM
Juniper Networks — vSRX-Firewall der nächsten Generation	Sendet Ergebnisse über Security Hub CSPM
k9 Security — Zugriffsanalysator	Sendet Ergebnisse über Security Hub CSPM
Schnürsenkelarbeit	Sendet Ergebnisse über Security Hub CSPM
McAfee — MVISION CNAPP	Sendet Ergebnisse über Security Hub CSPM
NETSCOUT — Cyber-Ermittler	Sendet Ergebnisse über Security Hub CSPM
Orca — Cloud-Sicherheitsplattform	Sendet Ergebnisse über Security Hub CSPM
Palo Alto Networks — Prisma Cloud Compute	Sendet Ergebnisse über Security Hub CSPM
Palo Alto Networks — Prisma Cloud Enterprise	Sendet Ergebnisse über Security Hub CSPM

Partner	Integration
Plerion — Cloud-Sicherheitsplattform	Sendet Ergebnisse über Security Hub CSPM
Prowler	Sendet Ergebnisse über Security Hub CSPM
Qualys — Schwachstellenmanagement	Sendet Ergebnisse über Security Hub CSPM
Rapid7 — InsightVM	Sendet Ergebnisse über Security Hub CSPM
SentinelOne	Sendet Ergebnisse über Security Hub CSPM
Snyk	Sendet Ergebnisse über Security Hub CSPM
Sonrai Security — Sonrai Dig	Sendet Ergebnisse über Security Hub CSPM
Sophos — Serverschutz	Sendet Ergebnisse über Security Hub CSPM
StackRox — Kubernetes-Sicherheit	Sendet Ergebnisse über Security Hub CSPM
Sumo Logic — Analyse von Maschinendaten	Sendet Ergebnisse über Security Hub CSPM
Symantec — Schutz von Cloud-Workloads	Sendet Ergebnisse über Security Hub CSPM
Tenable.io	Sendet Ergebnisse über Security Hub CSPM
Trend Micro — Cloud One	Sendet Ergebnisse über Security Hub CSPM

Partner	Integration
Vectra — Cognito Detect	Sendet Ergebnisse über Security Hub CSPM
Wiz	Sendet Ergebnisse über Security Hub CSPM
Caveonix — Caveonix Cloud	Sendet und empfängt Ergebnisse über Security Hub CSPM
Cloud Custodian	Sendet und empfängt Ergebnisse über Security Hub CSPM
DisruptOps	Sendet und empfängt Ergebnisse über Security Hub CSPM
Kion	Sendet und empfängt Ergebnisse über Security Hub CSPM
Turbot	Sendet und empfängt Ergebnisse über Security Hub CSPM

 Note

Diese Liste spiegelt die Security Hub-Partnerintegrationen wider, die zum Zeitpunkt der Erstellung dieses Artikels Ergebnisse senden. Da AWS Security Hub regelmäßig neue Partnerintegrationen hinzufügt, finden Sie im Security Hub-Benutzerhandbuch unter Produktintegrationen von Drittanbietern mit AWS Security Hub [CSPM](#) eine vollständige up-to-date Liste der verfügbaren Partner.

Exportieren von Protokolldaten nach Amazon S3

In diesem Kapitel finden Sie Informationen, damit Sie Protokolldaten aus Ihren Protokollgruppen in einen Amazon S3 S3-Bucket exportieren können, um sie individuell zu verarbeiten und zu analysieren oder auf andere Systeme zu laden. Sie können in einen S3-Bucket desselben Kontos oder eines anderen Kontos exportieren.

Sie haben die folgenden Möglichkeiten:

- Exportieren Sie Protokolldaten in S3-Buckets, die mit SSE-KMS in () verschlüsselt sind AWS Key Management Service AWS KMS
- Exportieren von Protokolldaten an S3-Buckets, die eine S3-Objektsperre mit einem Aufbewahrungszeitraum aktiviert haben

Wir empfehlen, dass Sie nicht regelmäßig nach Amazon S3 exportieren, um Ihre Protokolle kontinuierlich zu archivieren. Für diesen Anwendungsfall empfehlen wir stattdessen, Abonnements zu verwenden. Weitere Informationen zu Abonnements finden Sie unter [Echtzeitverarbeitung von Protokolldaten mit Abonnements](#).

Um den Exportprozess zu beginnen, müssen Sie einen S3-Bucket zum Speichern der exportierten Protokolldaten erstellen. Sie können die exportierten Dateien im S3-Bucket speichern und Amazon-S3-Lebenszyklusregeln definieren, mit denen die exportierten Dateien automatisch archiviert oder gelöscht werden.

Sie können in S3-Buckets exportieren, die mit AES-256 oder SSE-KMS verschlüsselt sind. Der Export in Buckets, die mit DSSE-KMS verschlüsselt sind, wird nicht unterstützt.

Sie können Protokolle aus mehreren Protokollgruppen oder mehreren Zeitbereichen in den gleichen S3-Bucket exportieren. Um exportierte Daten zu organisieren, geben Sie für jede Exportaufgabe ein Präfix an, das als Amazon S3 S3-Schlüsselpräfix für alle exportierten Objekte verwendet wird. Zum Beispiel `prod/app-logs/2026-01-03/` oder `log-group-name/backup/`

Note

Die zeitbasierte Sortierung von Protokolldaten in einer exportierten Datei wird nicht garantiert. Sie können die exportierten Protokollfelddaten mit Hilfe von Linux-Dienstprogrammen sortieren. Der folgende Hilfsprogrammbefehl sortiert beispielsweise die Ereignisse in allen Dateien vom Typ `.gz` in einem einzelnen Ordner.

```
find . -exec zcat {} + | sed -r 's/^[0-9]+/\x0&/' | sort -z
```

Der folgende Hilfsprogrammbefehl sortiert GZ-Dateien aus mehreren Unterordnern.

```
find ./ */ -type f -exec zcat {} + | sed -r 's/^[0-9]+/\x0&/' | sort -z
```

Darüber hinaus können Sie einen weiteren Befehl vom Typ `stdout` verwenden, um die sortierte Ausgabe zur Speicherung an eine andere Datei weiterzuleiten.

Protokolldaten können bis zu 12 Stunden für den Export verfügbar sein. Für Exportaufgaben tritt nach 24 Stunden ein Timeout auf. Falls bei Ihren Exportaufgaben ein Timeout auftritt, verringern Sie den Zeitraum, wenn Sie die Exportaufgabe erstellen.

Informationen zu Quasi-Echtzeit-Analysen von Protokolldaten finden Sie hingegen unter [Logdaten mit CloudWatch Logs Insights analysieren](#) oder [Echtzeitverarbeitung von Protokolldaten mit Abonnements](#).

Inhalt

- [Konzepte](#)
- [Exportieren von Protokolldaten in Amazon S3 mit der Konsole](#)
- [Exportieren Sie Protokolldaten nach Amazon S3 mit dem AWS CLI](#)
- [Exportaufgaben beschreiben \(CLI\)](#)
- [Eine Exportaufgabe abbrechen \(CLI\)](#)

Konzepte

Bevor Sie beginnen, sollten Sie sich mit folgenden Exportkonzepten vertraut machen:

Name der Protokollgruppe

Der Name der Protokollgruppe für einen Exportvorgang. Die Protokolldaten in dieser Protokollgruppe werden in den angegebenen S3-Bucket exportiert.

von (Zeitstempel)

Ein erforderlicher Zeitstempel in Millisekunden seit dem 1. Januar 1970 00:00:00 UTC. Alle Protokollereignisse in der Protokollgruppe, die an oder nach diesem Zeitpunkt aufgenommen wurden, werden exportiert.

bis (Zeitstempel)

Ein erforderlicher Zeitstempel in Millisekunden seit dem 1. Januar 1970 00:00:00 UTC. Alle Protokollereignisse in der Protokollgruppe, die vor diesem Zeitpunkt aufgenommen wurden, werden exportiert.

Ziel-Bucket

Der Name des mit einem Exportvorgang verknüpften S3-Bucket. Dieses Bucket dient zum Exportieren von Protokolldaten aus der angegebenen Protokollgruppe.

Ziel-Präfix

Ein optionales Attribut, das als Amazon-S3-Schlüsselpräfix für alle exportierten Objekte verwendet wird. Dadurch können Sie eine Art Ordnerstruktur in Ihrem Bucket erstellen.

Exportieren von Protokolldaten in Amazon S3 mit der Konsole

In den folgenden Beispielen verwenden Sie die CloudWatch Amazon-Konsole, um alle Daten aus einer Amazon CloudWatch Logs-Protokollgruppe mit dem Namen in einen Amazon S3-Bucket mit dem Namen `my-log-group` zu exportieren `amazn-s3-demo-bucket`.

Das Exportieren von Protokolldaten, die mit SSE-KMS verschlüsselt sind, in S3-Buckets wird unterstützt. Der Export in Buckets, die mit DSSE-KMS verschlüsselt sind, wird nicht unterstützt.

Wie Sie den Export im Detail einrichten, hängt davon ab, ob sich der Amazon-S3-Bucket, in den Sie exportieren möchten, im selben Konto wie die zu exportierenden Protokolle befindet oder in einem anderen Konto.

Themen

- [Export für dasselbe Konto \(Konsole\)](#)
- [Kontoübergreifender Export \(Konsole\)](#)

Export für dasselbe Konto (Konsole)

Wenn sich der Amazon-S3-Bucket im selben Konto befindet wie die zu exportierenden Protokolle, verwenden Sie die Anweisungen in diesem Abschnitt.

Themen

- [Erstellen Sie einen Amazon S3 S3-Bucket \(Konsole\)](#)
- [Richten Sie Zugriffsberechtigungen ein \(Konsole\)](#)
- [Berechtigungen für einen Amazon S3 S3-Bucket \(Konsole\) festlegen](#)
- [\(Optional\) Exportieren in einen Amazon S3 S3-Ziel-Bucket, der mit SSE-KMS verschlüsselt ist \(Konsole\)](#)
- [Erstellen Sie eine Exportaufgabe \(Konsole\)](#)

Erstellen Sie einen Amazon S3 S3-Bucket (Konsole)

Wir empfehlen Ihnen, einen Bucket zu verwenden, der speziell für CloudWatch Logs erstellt wurde. Wenn Sie jedoch einen vorhandenen Bucket verwenden möchten, gehen Sie direkt zu Schritt 2.

Note

Der Amazon-S3-Bucket muss sich in derselben Region wie die Protokolldaten befinden, die exportiert werden sollen. CloudWatch Logs unterstützt den Export von Daten in Amazon S3 S3-Buckets in einer anderen Region nicht.

So erstellen Sie einen Amazon-S3-Bucket

1. Öffnen Sie die Amazon S3 S3-Konsole unter <https://console.aws.amazon.com/s3/>.
2. Ändern Sie, falls erforderlich, die Region. Wählen Sie in der Navigationsleiste die Region aus, in der sich Ihre CloudWatch Logs befinden.
3. Wählen Sie Create Bucket (Bucket erstellen) aus.
4. Geben Sie unter Bucket-Name einen Namen für den Bucket ein.
5. Wählen Sie unter Region die Region aus, in der sich Ihre CloudWatch Logs-Daten befinden.
6. Wählen Sie Erstellen aus.

Richten Sie Zugriffsberechtigungen ein (Konsole)

Um die Exportaufgabe zu erstellen, müssen Sie mit der `AmazonS3ReadOnlyAccess` IAM-Rolle angemeldet sein und über die folgenden Berechtigungen verfügen:

- `logs:CreateExportTask`
- `logs:CancelExportTask`
- `logs:DescribeExportTasks`
- `logs:DescribeLogStreams`
- `logs:DescribeLogGroups`

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter [Erstellen eines Berechtigungssatzes](#) im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter [Eine Rolle für einen externen Identitätsanbieter \(Verbund\) erstellen](#) im IAM-Benutzerhandbuch.

- IAM-Benutzer:

- Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter [Eine Rolle für einen IAM-Benutzer erstellen](#) im IAM-Benutzerhandbuch.
- (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter [Hinzufügen von Berechtigungen zu einem Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

Berechtigungen für einen Amazon S3 S3-Bucket (Konsole) festlegen

Standardmäßig werden alle Amazon-S3-Buckets und -Objekte als privat eingestuft. Nur der Besitzer der Ressource AWS-Konto, der den Bucket erstellt hat, kann auf den Bucket und alle darin enthaltenen Objekte zugreifen. Der Ressourcenbesitzer kann jedoch anderen Ressourcen und Benutzern Zugriffsberechtigungen gewähren, indem er eine Zugriffsrichtlinie schreibt.

Wenn Sie die Richtlinie festlegen, empfehlen wir Ihnen, eine zufällig generierte Zeichenfolge als Präfix für den Bucket einzufügen, so dass nur die beabsichtigten Protokoll-Streams in den Bucket exportiert werden.

Important

Um Exporte in Amazon S3 S3-Buckets sicherer zu machen, müssen Sie jetzt die Liste der Quellkonten angeben, die Protokolldaten in Ihren S3-Bucket exportieren dürfen.

Im folgenden Beispiel entspricht die Liste der Konten IDs im `aws:SourceAccount` Schlüssel den Konten, von denen ein Benutzer Protokolldaten in Ihren Amazon S3 S3-Bucket exportieren kann. Der Schlüssel `aws:SourceArn` ist die Ressource, für die die Aktion ausgeführt wird. Sie können dies auf eine bestimmte Protokollgruppe beschränken oder einen Platzhalter verwenden, wie in diesem Beispiel zu sehen.

Wir empfehlen, dass Sie auch die Konto-ID des Kontos angeben, in dem der S3-Bucket erstellt wurde, um den Export innerhalb desselben Kontos zu ermöglichen.

So legen Sie Berechtigungen für einen Amazon-S3-Bucket fest

1. Wählen Sie in der Amazon S3 S3-Konsole den Bucket aus, den Sie erstellt haben.
2. Klicken Sie auf Permissions (Berechtigungen), Bucket policy (Bucket-Richtlinie).
3. Fügen Sie im Bucket policy editor (Bucket-Richtlinieneditor) die folgende Richtlinie hinzu. Ändern Sie `amzn-s3-demo-bucket` in den Namen Ihres S3-Bucket. Achten Sie auf die Angabe des korrekten Regionsendpunkts (beispielsweise `us-west-1`) für Prinzipal.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCloudWatchLogsGetBucketAcl",
      "Action": "s3:GetBucketAcl",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
      "Condition": {
        "StringEquals": {
```

```

        "aws:SourceAccount": [
            "123456789012",
            "111122223333"
        ],
    },
    "ArnLike": {
        "aws:SourceArn": [
            "arn:aws:logs:us-east-1:123456789012:log-group:*",
            "arn:aws:logs:us-east-1:111122223333:log-group:*"
        ]
    }
},
{
    "Sid": "AllowCloudWatchLogsPutObject",
    "Action": "s3:PutObject",
    "Effect": "Allow",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
    "Condition": {
        "StringEquals": {
            "s3:x-amz-acl": "bucket-owner-full-control",
            "aws:SourceAccount": [
                "123456789012",
                "111122223333"
            ]
        },
        "ArnLike": {
            "aws:SourceArn": [
                "arn:aws:logs:us-east-1:123456789012:log-group:*",
                "arn:aws:logs:us-east-1:111122223333:log-group:*"
            ]
        }
    }
}
]
}

```

4. Wählen Sie **Save** aus, um die Richtlinie, die Sie gerade hinzugefügt haben, als Zugriffsrichtlinie für den Bucket festzulegen. Diese Richtlinie ermöglicht CloudWatch Logs, Protokolldaten in Ihren Amazon S3 S3-Bucket zu exportieren. Der Bucket-Eigentümer hat vollen Zugriff auf alle exportierten Objekte.

 Warning

Wenn mit dem vorhandenen Bucket bereits eine oder mehrere Richtlinien verknüpft sind, fügen Sie die Anweisungen für den Zugriff auf CloudWatch Logs zu dieser Richtlinie oder diesen Richtlinien hinzu. Sie sollten eine Beurteilung der daraus resultierenden Berechtigungen vornehmen, um sicherzustellen, dass sie für die Benutzer, die auf den Bucket zugreifen werden, geeignet sind.

(Optional) Exportieren in einen Amazon S3 S3-Ziel-Bucket, der mit SSE-KMS verschlüsselt ist (Konsole)

Dieser Schritt ist nur erforderlich, wenn Sie in einen Amazon S3 S3-Bucket exportieren, der serverseitige Verschlüsselung mit AWS KMS keys verwendet. Diese Verschlüsselung wird als SSE-KMS bezeichnet.

So exportieren Sie in einen mit SSE-KMS verschlüsselten Bucket

1. Öffnen Sie die AWS KMS Konsole unter <https://console.aws.amazon.com/kms>.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie in linken Navigationsleiste auf Customer managed keys (Vom Kunden verwaltete Schlüssel).

Klicken Sie auf Create key (Schlüssel erstellen).

4. Wählen Sie für Key type (Schlüsseltyp) Symmetric (Symmetrisch).
5. Wählen Sie für Key usage (Schlüsselverwendung) die Option Encrypt and decrypt (Verschlüsseln und Entschlüsseln) und dann Next (Weiter) aus.
6. Geben Sie unter Add Labels (Bezeichnungen hinzufügen) einen Alias für den Schlüssel ein und fügen Sie optional eine Beschreibung oder Tags hinzu. Klicken Sie anschließend auf Weiter.
7. Wählen Sie unter Key administrators (Schlüsseladministratoren) aus, wer diesen Schlüssel verwalten kann, und klicken Sie dann auf Next (Weiter).
8. Nehmen Sie unter Define key usage permissions (Schlüsselverwendungsberechtigungen definieren) keine Änderungen vor und wählen Sie Next (Weiter) aus.
9. Überprüfen Sie die Einstellungen und klicken Sie anschließend auf Finish (Fertig).

10. Wählen Sie auf der Seite Customer managed keys (Vom Kunden verwaltete Schlüssel) den Namen des Schlüssels aus, den Sie gerade erstellt haben.
11. Wählen Sie den Reiter Key policy (Schlüsselrichtlinie) und Switch to policy view (Zur Richtlinienansicht wechseln) aus.
12. Wählen Sie im Abschnitt Key policy (Schlüsselrichtlinie) die Option Edit (Bearbeiten) aus.
13. Fügen Sie der Anweisungsliste der Schlüsselrichtlinie die folgende Anweisung hinzu. Wenn Sie dies tun, *Region* ersetzen Sie es durch die Region Ihrer Protokolle und *account-ARN* ersetzen Sie es durch den ARN des Kontos, dem der KMS-Schlüssel gehört.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow CWL Service Principal usage",
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.Region.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "account-ARN"
      },
      "Action": [
        "kms:GetKeyPolicy*",
        "kms:PutKeyPolicy*",
        "kms:DescribeKey*",
        "kms:CreateAlias*",
        "kms:ScheduleKeyDeletion*",
        "kms:Decrypt"
      ],
      "Resource": "*"
    }
  ]
}
```

```
}  
]  
}
```

14. Wählen Sie Änderungen speichern aus.
15. Öffnen Sie die Amazon S3 S3-Konsole unter <https://console.aws.amazon.com/s3/>.
16. Wählen Sie den Bucket, den Sie in [Erstellen Sie einen S3-Bucket \(CLI\)](#) erstellt haben, und wählen Sie den Bucket-Namen aus.
17. Wählen Sie die Registerkarte Eigenschaften aus. Wählen Sie dann unter Default encryption (Standard-Verschlüsselung) Edit (Bearbeiten) aus.
18. Unter den Optionen für Server-side encryption (Serverseitige Verschlüsselung) wählen Sie Enable (Aktivieren) aus.
19. Wählen Sie unter Encryption type (Verschlüsselungs-Typ) AWS Key Management Service - Schlüssel (SSE-KMS) aus.
20. Wählen Sie aus Ihren AWS KMS Schlüsseln und suchen Sie den Schlüssel, den Sie erstellt haben.
21. Wählen Sie unter Bucket Key (Bucket-Schlüssel) Enable (Aktivieren) aus.
22. Wählen Sie Änderungen speichern aus.

Erstellen Sie eine Exportaufgabe (Konsole)

In diesem Verfahren erstellen Sie die Exportaufgabe für den Export von Protokollen aus einer Protokollgruppe.

Um Daten mit der CloudWatch Konsole nach Amazon S3 zu exportieren

1. Melden Sie sich mit ausreichenden Berechtigungen an, wie unter [Richten Sie Zugriffsberechtigungen ein \(Konsole\)](#) dokumentiert.
2. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
3. Wählen Sie im Navigationsbereich Protokollgruppen aus.
4. Wählen Sie im Bildschirm Protokollgruppen den Namen der Protokollgruppe aus.
5. Wählen Sie für Actions (Aktionen) die Option Export data to Amazon S3 (Daten nach Amazon S3 exportieren) aus.

6. Legen Sie im Bildschirm Export data to Amazon S3 (Daten nach Amazon S3 exportieren) unter Define data export (Datenexport definieren) den Zeitraum für die zu exportierenden Daten mit From (Von) und To (Bis) fest.
7. Wenn Ihre Protokollgruppe über mehrere Protokoll-Streams verfügt, können Sie ein Protokoll-Stream-Präfix angeben, um die Protokollgruppendaten an einen bestimmten Stream zu beschränken. Wählen Sie Advanced (Erweitert) aus und geben Sie für Stream prefix (Stream-Präfix) das Protokoll-Stream-Präfix ein.
8. Wählen Sie unter Choose S3 bucket (S3-Bucket auswählen) das Konto für den S3-Bucket aus.
9. Wählen Sie für S3 bucket name einen S3-Bucket aus.
10. Geben Sie für S3-Bucket-Präfix die zufällig generierte Zeichenfolge ein, die Sie in der Bucket-Richtlinie angegeben haben.
11. Wählen Sie Export (Exportieren) aus, um Ihre Protokolldaten nach Amazon S3 zu exportieren.
12. Um den Status der Protokolldaten anzuzeigen, die Sie in Amazon S3 exportiert haben, wählen Sie Actions (Aktionen) und dann View all exports to Amazon S3 (Alle Exporte in Amazon S3 anzeigen) aus.

Kontoübergreifender Export (Konsole)

Wenn sich der Amazon-S3-Bucket in einem anderen Konto befindet als die zu exportierenden Protokolle, verwenden Sie die Anweisungen in diesem Abschnitt.

Themen

- [Einen Amazon S3 S3-Bucket für den kontoübergreifenden Export erstellen \(Konsole\)](#)
- [Richten Sie Zugriffsberechtigungen für den kontoübergreifenden Export ein \(Konsole\)](#)
- [Legen Sie Berechtigungen für einen S3-Bucket für den kontoübergreifenden Export fest \(Konsole\)](#)
- [\(Optional\) Exportieren in einen Amazon S3 S3-Ziel-Bucket, der mit SSE-KMS für den kontoübergreifenden Export verschlüsselt ist \(Konsole\)](#)
- [Erstellen Sie eine Exportaufgabe für den kontoübergreifenden Export \(Konsole\)](#)

Einen Amazon S3 S3-Bucket für den kontoübergreifenden Export erstellen (Konsole)

Wir empfehlen Ihnen, einen Bucket zu verwenden, der speziell für CloudWatch Logs erstellt wurde. Wenn Sie jedoch einen vorhandenen Bucket verwenden möchten, können Sie dieses Verfahren überspringen.

 Note

Der Amazon-S3-Bucket muss sich in derselben Region wie die Protokolldaten befinden, die exportiert werden sollen. CloudWatch Logs unterstützt den Export von Daten in Amazon S3 S3-Buckets in einer anderen Region nicht.

So erstellen Sie einen Amazon-S3-Bucket

1. Öffnen Sie die Amazon S3 S3-Konsole unter <https://console.aws.amazon.com/s3/>.
2. Ändern Sie, falls erforderlich, die Region. Wählen Sie in der Navigationsleiste die Region aus, in der sich Ihre CloudWatch Logs befinden.
3. Wählen Sie Create Bucket (Bucket erstellen) aus.
4. Geben Sie unter Bucket-Name einen Namen für den Bucket ein.
5. Wählen Sie unter Region die Region aus, in der sich Ihre CloudWatch Logs-Daten befinden.
6. Wählen Sie Erstellen aus.

Richten Sie Zugriffsberechtigungen für den kontoübergreifenden Export ein (Konsole)

Zunächst müssen Sie eine neue IAM-Richtlinie erstellen, damit CloudWatch Logs die `s3:PutObject` Aktion für den Amazon S3 S3-Ziel-Bucket im Zielkonto hat.

Neben der `s3:PutObject` Aktion hängen weitere in der Richtlinie enthaltene Aktionen davon ab, ob der Ziel-Bucket AWS KMS Verschlüsselung verwendet oder mit der Einstellung [S3-Objektbesitz ACLs](#) aktiviert wurde.

- Wenn Sie die KMS-Verschlüsselung verwenden, fügen Sie die `kms:Decrypt` Aktionen `kms:GenerateDataKey` und für die Schlüsselressource hinzu
- Wenn sie im Bucket aktiviert ACLs sind, fügen Sie die `s3:PutObjectAc1` Aktion für die Bucket-Ressource hinzu

Ändern amzn-s3-demo-bucket Sie in den folgenden Richtlinien den Namen Ihres Ziel-S3-Buckets.

So erstellen Sie eine IAM-Richtlinie für den Export von Protokollen in einen Amazon-S3-Bucket

1. Öffnen Sie unter <https://console.aws.amazon.com/iam/> die IAM-Konsole.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien).

3. Wählen Sie Richtlinie erstellen aus.
4. Wählen Sie im Abschnitt Richtlinien-Editor JSON aus.
5. Wenn der Ziel-Bucket keine AWS KMS Verschlüsselung verwendet, fügen Sie die folgende Richtlinie in den Editor ein.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
  ]
}
```

Wenn der Ziel-Bucket AWS KMS Verschlüsselung verwendet, fügen Sie die folgende Richtlinie in den Editor ein.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:us-east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ]
}
```

```
}  
]  
}
```

Wenn sie im Ziel-Bucket aktiviert ACLs sind, fügen Sie in den obigen Richtlinien den Block s3: PutObjectAcl zum Block s3: PutObject Action hinzu.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:PutObject",  
        "s3:PutObjectAcl"  
      ],  
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"  
    }  
  ]  
}
```

6. Wählen Sie Weiter aus.
7. Geben Sie den Namen einer Richtlinie ein. Sie verwenden diesen Namen, um die Richtlinie an Ihre IAM-Rolle anzuhängen.
8. Wählen Sie Richtlinie erstellen aus, um die neue Richtlinie zu speichern.

Um eine Exportaufgabe zu erstellen, müssen Sie mit einer IAM-Rolle angemeldet sein, der die AmazonS3ReadOnlyAccess verwaltete Richtlinie und die oben erstellte IAM-Richtlinie zugeordnet sind. Außerdem müssen Sie über die folgenden Berechtigungen verfügen:

- logs:CreateExportTask
- logs:CancelExportTask
- logs:DescribeExportTasks
- logs:DescribeLogStreams
- logs:DescribeLogGroups

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in: AWS IAM Identity Center

Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter [Erstellen eines Berechtigungssatzes](#) im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter [Eine Rolle für einen externen Identitätsanbieter \(Verbund\) erstellen](#) im IAM-Benutzerhandbuch.

- IAM-Benutzer:

- Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter [Eine Rolle für einen IAM-Benutzer erstellen](#) im IAM-Benutzerhandbuch.
- (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter [Hinzufügen von Berechtigungen zu einem Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

Legen Sie Berechtigungen für einen S3-Bucket für den kontoübergreifenden Export fest (Konsole)

Standardmäßig werden alle S3-Buckets und -Objekte als privat eingestuft. Nur der Ressourcenbesitzer AWS-Konto, der den Bucket erstellt hat, kann auf den Bucket und alle darin enthaltenen Objekte zugreifen. Der Ressourcenbesitzer kann jedoch anderen Ressourcen und Benutzern Zugriffsberechtigungen gewähren, indem er eine Zugriffsrichtlinie schreibt.

Wenn Sie die Richtlinie festlegen, empfehlen wir Ihnen, eine zufällig generierte Zeichenfolge als Präfix für den Bucket einzufügen, so dass nur die beabsichtigten Protokoll-Streams in den Bucket exportiert werden.

Important

Um Exporte in S3-Buckets sicherer zu machen, müssen Sie jetzt die Liste der Quellkonten angeben, die Protokolldaten in Ihren S3-Bucket exportieren dürfen.

Im folgenden Beispiel entspricht die Liste der Konten IDs im `aws:SourceAccount` Schlüssel den Konten, von denen ein Benutzer Protokolldaten in Ihren S3-Bucket exportieren kann. Der Schlüssel `aws:SourceArn` ist die Ressource, für die die Aktion ausgeführt wird. Sie können

dies auf eine bestimmte Protokollgruppe beschränken oder einen Platzhalter verwenden, wie in diesem Beispiel zu sehen.

Wir empfehlen, dass Sie auch die Konto-ID des Kontos angeben, in dem der S3-Bucket erstellt wurde, um den Export innerhalb desselben Kontos zu ermöglichen.

So legen Sie Berechtigungen für einen Amazon-S3-Bucket fest

1. Wählen Sie in der Amazon S3 S3-Konsole den Bucket aus, den Sie erstellt haben.
2. Klicken Sie auf Permissions (Berechtigungen), Bucket policy (Bucket-Richtlinie).
3. Fügen Sie im Bucket policy editor (Bucket-Richtlinieneditor) die folgende Richtlinie hinzu. Ändern Sie `amzn-s3-demo-bucket` in den Namen Ihres S3-Bucket. Achten Sie auf die Angabe des korrekten Regionsendpunkts (beispielsweise `us-east-1`) für Prinzipal.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "s3:GetBucketAcl",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": [
            "123456789012",
            "111122223333"
          ]
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:logs:us-east-1:123456789012:log-group:*",
            "arn:aws:logs:us-east-1:111122223333:log-group:*"
          ]
        }
      }
    }
  ],
}
```



```

{
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
  "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
  "Condition": {
    "StringEquals": {
      "s3:x-amz-acl": "bucket-owner-full-control",
      "aws:SourceAccount": [
        "123456789012",
        "111122223333"
      ]
    },
    "ArnLike": {
      "aws:SourceArn": [
        "arn:aws:logs:us-east-1:123456789012:log-group:*",
        "arn:aws:logs:us-east-1:111122223333:log-group:*"
      ]
    }
  }
},
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/role_name"
  },
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
  "Condition": {
    "StringEquals": {
      "s3:x-amz-acl": "bucket-owner-full-control"
    }
  }
}
]
}

```

4. Wählen Sie **Save** aus, um die Richtlinie, die Sie gerade hinzugefügt haben, als Zugriffsrichtlinie für den Bucket festzulegen. Diese Richtlinie ermöglicht CloudWatch Logs, Protokolldaten in Ihren S3-Bucket zu exportieren. Der Bucket-Eigentümer hat vollen Zugriff auf alle exportierten Objekte.

 Warning

Wenn mit dem vorhandenen Bucket bereits eine oder mehrere Richtlinien verknüpft sind, fügen Sie die Anweisungen für den Zugriff auf CloudWatch Logs zu dieser Richtlinie oder diesen Richtlinien hinzu. Sie sollten eine Beurteilung der daraus resultierenden Berechtigungen vornehmen, um sicherzustellen, dass sie für die Benutzer, die auf den Bucket zugreifen werden, geeignet sind.

(Optional) Exportieren in einen Amazon S3 S3-Ziel-Bucket, der mit SSE-KMS für den kontoübergreifenden Export verschlüsselt ist (Konsole)

Dieses Verfahren ist nur erforderlich, wenn Sie in einen S3-Bucket exportieren, der serverseitige Verschlüsselung mit verwendet. AWS KMS keys Diese Verschlüsselung wird als SSE-KMS bezeichnet.

So exportieren Sie in einen mit SSE-KMS verschlüsselten Bucket

1. Öffnen Sie die AWS KMS Konsole unter <https://console.aws.amazon.com/kms>.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie in linken Navigationsleiste auf Customer managed keys (Vom Kunden verwaltete Schlüssel).

Klicken Sie auf Create key (Schlüssel erstellen).

4. Wählen Sie für Key type (Schlüsseltyp) Symmetric (Symmetrisch).
5. Wählen Sie für Key usage (Schlüsselverwendung) die Option Encrypt and decrypt (Verschlüsseln und Entschlüsseln) und dann Next (Weiter) aus.
6. Geben Sie unter Add Labels (Bezeichnungen hinzufügen) einen Alias für den Schlüssel ein und fügen Sie optional eine Beschreibung oder Tags hinzu. Klicken Sie anschließend auf Weiter.
7. Wählen Sie unter Key administrators (Schlüsseladministratoren) aus, wer diesen Schlüssel verwalten kann, und klicken Sie dann auf Next (Weiter).
8. Nehmen Sie unter Define key usage permissions (Schlüsselverwendungsberechtigungen definieren) keine Änderungen vor und wählen Sie Next (Weiter) aus.
9. Überprüfen Sie die Einstellungen und klicken Sie anschließend auf Finish (Fertig).

10. Wählen Sie auf der Seite Customer managed keys (Vom Kunden verwaltete Schlüssel) den Namen des Schlüssels aus, den Sie gerade erstellt haben.
11. Wählen Sie den Reiter Key policy (Schlüsselrichtlinie) und Switch to policy view (Zur Richtlinienansicht wechseln) aus.
12. Wählen Sie im Abschnitt Key policy (Schlüsselrichtlinie) die Option Edit (Bearbeiten) aus.
13. Fügen Sie der Anweisungsliste der Schlüsselrichtlinie die folgende Anweisung hinzu. Wenn Sie dies tun, *us-east-1* ersetzen Sie es durch die Region Ihrer Protokolle, *account-ARN* durch den ARN des Kontos, dem der KMS-Schlüssel gehört, durch die Kontonummer, *123456789012* dem der KMS-Schlüssel gehört, *key_id* durch die KMS-Schlüssel-ID und *role_name* durch die Rolle, die für die Erstellung der Exportaufgabe verwendet wurde.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow CWL Service Principal usage",
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "account-ARN"
      },
      "Action": [
        "kms:GetKeyPolicy*",
        "kms:PutKeyPolicy*",
        "kms:DescribeKey*",
        "kms:CreateAlias*",
        "kms:ScheduleKeyDeletion*",

```

```

        "kms:Decrypt"
    ],
    "Resource": "*"
  },
  {
    "Sid": "Enable IAM Role Permissions",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::111122223333:role/role_name"
    },
    "Action": [
      "kms:GenerateDataKey",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-id"
  }
]
}

```

14. Wählen Sie Änderungen speichern aus.
15. Öffnen Sie die Amazon S3 S3-Konsole unter <https://console.aws.amazon.com/s3/>.
16. Wählen Sie den Bucket, den Sie in [Erstellen Sie einen S3-Bucket \(CLI\)](#) erstellt haben, und wählen Sie den Bucket-Namen aus.
17. Wählen Sie die Registerkarte Eigenschaften aus. Wählen Sie dann unter Default encryption (Standard-Verschlüsselung) Edit (Bearbeiten) aus.
18. Unter den Optionen für Server-side encryption (Serverseitige Verschlüsselung) wählen Sie Enable (Aktivieren) aus.
19. Wählen Sie unter Encryption type (Verschlüsselungs-Typ) AWS Key Management Service - Schlüssel (SSE-KMS) aus.
20. Wählen Sie aus Ihren AWS KMS Schlüsseln und suchen Sie den Schlüssel, den Sie erstellt haben.
21. Wählen Sie unter Bucket Key (Bucket-Schlüssel) Enable (Aktivieren) aus.
22. Wählen Sie Änderungen speichern aus.

Erstellen Sie eine Exportaufgabe für den kontoübergreifenden Export (Konsole)

In diesem Verfahren erstellen Sie die Exportaufgabe für den Export von Protokollen aus einer Protokollgruppe.

Um Daten mit der CloudWatch Konsole nach Amazon S3 zu exportieren

1. Melden Sie sich mit ausreichenden Berechtigungen an, wie unter [Richten Sie Zugriffsberechtigungen ein \(Konsole\)](#) dokumentiert.
2. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
3. Wählen Sie im Navigationsbereich Protokollgruppen aus.
4. Wählen Sie im Bildschirm Protokollgruppen den Namen der Protokollgruppe aus.
5. Wählen Sie für Actions (Aktionen) die Option Export data to Amazon S3 (Daten nach Amazon S3 exportieren) aus.
6. Legen Sie im Bildschirm Export data to Amazon S3 (Daten nach Amazon S3 exportieren) unter Define data export (Datenexport definieren) den Zeitraum für die zu exportierenden Daten mit From (Von) und To (Bis) fest.
7. Wenn Ihre Protokollgruppe über mehrere Protokoll-Streams verfügt, können Sie ein Protokoll-Stream-Präfix angeben, um die Protokollgruppendaten an einen bestimmten Stream zu beschränken. Wählen Sie Advanced (Erweitert) aus und geben Sie für Stream prefix (Stream-Präfix) das Protokoll-Stream-Präfix ein.
8. Wählen Sie unter Choose S3 bucket (S3-Bucket auswählen) das Konto für den S3-Bucket aus.
9. Wählen Sie für S3 bucket name (Name des S3-Bucket) einen S3-Bucket aus.
10. Geben Sie für S3-Bucket-Präfix die zufällig generierte Zeichenfolge ein, die Sie in der Bucket-Richtlinie angegeben haben.
11. Wählen Sie Export (Exportieren) aus, um Ihre Protokolldaten nach Amazon S3 zu exportieren.
12. Um den Status der Protokolldaten anzuzeigen, die Sie in Amazon S3 exportiert haben, wählen Sie Actions (Aktionen) und dann View all exports to Amazon S3 (Alle Exporte in Amazon S3 anzeigen) aus.

Exportieren Sie Protokolldaten nach Amazon S3 mit dem AWS CLI

Im folgenden Beispiel verwenden Sie eine Exportaufgabe, um alle Daten aus einer CloudWatch Logs-Protokollgruppe mit dem Namen in einen Amazon S3 S3-Bucket mit dem Namen my-log-group zu exportierenamzn-s3-demo-bucket. In diesem Beispiel wird davon ausgegangen, dass Sie bereits eine Protokollgruppe namens my-log-group erstellt haben.

Das Exportieren von Protokolldaten in S3-Buckets, die verschlüsselt sind, AWS KMS wird unterstützt. Der Export in Buckets, die mit DSSE-KMS verschlüsselt sind, wird nicht unterstützt.

Wie Sie den Export im Detail einrichten, hängt davon ab, ob sich der Amazon-S3-Bucket, in den Sie exportieren möchten, im selben Konto wie die zu exportierenden Protokolle befindet oder in einem anderen Konto.

Themen

- [Export für dasselbe Konto \(CLI\)](#)
- [Kontoübergreifender Export \(CLI\)](#)

Export für dasselbe Konto (CLI)

Wenn sich der Amazon-S3-Bucket im selben Konto befindet wie die zu exportierenden Protokolle, verwenden Sie die Anweisungen in diesem Abschnitt.

Themen

- [Erstellen Sie einen S3-Bucket \(CLI\)](#)
- [Zugriffsberechtigungen einrichten \(CLI\)](#)
- [Berechtigungen für einen S3-Bucket \(CLI\) festlegen](#)
- [\(Optional\) Exportieren in einen Amazon S3 S3-Ziel-Bucket, der mit SSE-KMS \(CLI\) verschlüsselt ist](#)
- [Erstellen Sie eine Exportaufgabe \(CLI\)](#)

Erstellen Sie einen S3-Bucket (CLI)

Wir empfehlen, dass Sie einen Bucket verwenden, der speziell für CloudWatch Logs erstellt wurde. Wenn Sie jedoch einen vorhandenen Bucket verwenden möchten, können Sie dieses Verfahren überspringen.

Note

Der S3-Bucket muss sich in derselben Region wie die Protokolldaten befinden, die exportiert werden sollen. CloudWatch Logs unterstützt den Export von Daten in S3-Buckets in einer anderen Region nicht.

Um einen S3-Bucket mit dem zu erstellen AWS CLI

Führen Sie an der Eingabeaufforderung den Befehl [create-bucket](#) aus, wobei LocationConstraint die Region ist, in der Sie Protokolldaten exportieren.

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --create-bucket-configuration  
LocationConstraint=us-east-2
```

Es folgt eine Beispielausgabe.

```
{  
  "Location": "/amzn-s3-demo-bucket"  
}
```

Zugriffsberechtigungen einrichten (CLI)

Um die Exportaufgabe später zu erstellen, müssen Sie mit der AmazonS3ReadOnlyAccess IAM-Rolle angemeldet sein und über die folgenden Berechtigungen verfügen:

- logs:CreateExportTask
- logs:CancelExportTask
- logs:DescribeExportTasks
- logs:DescribeLogStreams
- logs:DescribeLogGroups

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter [Erstellen eines Berechtigungssatzes](#) im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter [Eine Rolle für einen externen Identitätsanbieter \(Verbund\) erstellen](#) im IAM-Benutzerhandbuch.

- IAM-Benutzer:

- Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter [Eine Rolle für einen IAM-Benutzer erstellen](#) im IAM-Benutzerhandbuch.

- (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter [Hinzufügen von Berechtigungen zu einem Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

Berechtigungen für einen S3-Bucket (CLI) festlegen

Standardmäßig werden alle S3-Buckets und -Objekte als privat eingestuft. Nur der Ressourceneigentümer, das Konto, in dem der Bucket erstellt wurde, kann auf den Bucket und alle darin enthaltenen Objekte zugreifen. Der Ressourcenbesitzer kann jedoch anderen Ressourcen und Benutzern Zugriffsberechtigungen gewähren, indem er eine Zugriffsrichtlinie schreibt.

Important

Um Exporte in S3-Buckets sicherer zu machen, müssen Sie jetzt die Liste der Quellkonten angeben, die Protokolldaten in Ihren S3-Bucket exportieren dürfen.

Im folgenden Beispiel entspricht die Liste der Konten IDs im `aws:SourceAccount` Schlüssel den Konten, von denen ein Benutzer Protokolldaten in Ihren S3-Bucket exportieren kann. Der Schlüssel `aws:SourceArn` ist die Ressource, für die die Aktion ausgeführt wird. Sie können dies auf eine bestimmte Protokollgruppe beschränken oder einen Platzhalter verwenden, wie in diesem Beispiel zu sehen.

Wir empfehlen, dass Sie auch die Konto-ID des Kontos angeben, in dem der S3-Bucket erstellt wurde, um den Export innerhalb desselben Kontos zu ermöglichen.

So legen Sie Berechtigungen für einen S3-Bucket fest

1. Erstellen Sie eine Datei mit dem Namen `policy.json` und fügen Sie die folgende Zugriffsrichtlinie hinzu. Ändern Sie dabei `amzn-s3-demo-bucket` in den Namen Ihres S3-Bucket und `Principal` in den Endpunkt der Region, in der Sie Protokolldaten exportieren (beispielsweise `us-east-1`). Verwenden Sie einen Text-Editor, um diese Richtliniendatei zu erstellen. Verwenden Sie nicht die IAM-Konsole.

JSON

```
{  
  "Version": "2012-10-17",
```



```

"Statement": [
  {
    "Sid": "AllowGetBucketAcl",
    "Action": "s3:GetBucketAcl",
    "Effect": "Allow",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
    "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": [
          "123456789012",
          "111122223333"
        ]
      },
      "ArnLike": {
        "aws:SourceArn": [
          "arn:aws:logs:us-east-1:123456789012:log-group:*",
          "arn:aws:logs:us-east-1:111122223333:log-group:*"
        ]
      }
    }
  },
  {
    "Sid": "AllowPutObject",
    "Action": "s3:PutObject",
    "Effect": "Allow",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
    "Condition": {
      "StringEquals": {
        "s3:x-amz-acl": "bucket-owner-full-control",
        "aws:SourceAccount": [
          "123456789012",
          "111122223333"
        ]
      },
      "ArnLike": {
        "aws:SourceArn": [
          "arn:aws:logs:us-east-1:123456789012:log-group:*",
          "arn:aws:logs:us-east-1:111122223333:log-group:*"
        ]
      }
    }
  }
]

```

```
]
}
```

2. Legen Sie die Richtlinie, die Sie gerade hinzugefügt haben, mithilfe des [put-bucket-policy](#) Befehls als Zugriffsrichtlinie für Ihren Bucket fest. Diese Richtlinie ermöglicht CloudWatch Logs, Protokolldaten in Ihren S3-Bucket zu exportieren. Der Bucket-Eigentümer hat vollen Zugriff auf alle exportierten Objekte.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://
policy.json
```

 Warning

Wenn mit dem vorhandenen Bucket bereits eine oder mehrere Richtlinien verknüpft sind, fügen Sie die Anweisungen für den Zugriff auf CloudWatch Logs zu dieser Richtlinie oder diesen Richtlinien hinzu. Sie sollten eine Beurteilung der daraus resultierenden Berechtigungen vornehmen, um sicherzustellen, dass sie für die Benutzer, die auf den Bucket zugreifen werden, geeignet sind.

(Optional) Exportieren in einen Amazon S3 S3-Ziel-Bucket, der mit SSE-KMS (CLI) verschlüsselt ist

Dieses Verfahren ist nur erforderlich, wenn Sie in einen S3-Bucket exportieren, der serverseitige Verschlüsselung mit verwendet. AWS KMS keys Diese Verschlüsselung wird als SSE-KMS bezeichnet.

So exportieren Sie in einen mit SSE-KMS verschlüsselten Bucket

1. Verwenden Sie einen Texteditor, um eine Datei mit dem Namen `key_policy.json` zu erstellen und die folgende Zugriffsrichtlinie hinzuzufügen. Wenn Sie die Richtlinie hinzufügen, nehmen Sie die folgenden Änderungen vor:
 - **Region** Ersetzen Sie es durch die Region Ihrer Protokolle.
 - **account-ARN** Ersetzen Sie ihn durch den ARN des Kontos, dem der KMS-Schlüssel gehört.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow CWL Service Principal usage",
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.Region.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "account-ARN"
      },
      "Action": [
        "kms:GetKeyPolicy*",
        "kms:PutKeyPolicy*",
        "kms:DescribeKey*",
        "kms:CreateAlias*",
        "kms:ScheduleKeyDeletion*",
        "kms:Decrypt"
      ],
      "Resource": "*"
    }
  ]
}
```

2. Geben Sie den folgenden Befehl ein:

```
aws kms create-key --policy file://key_policy.json
```

Das Folgende ist eine Beispielausgabe aus diesem Befehl:

```
{
  "KeyMetadata": {
    "AWSAccountId": "account_id",
    "KeyId": "key_id",
    "Arn": "arn:aws:kms:us-east-2:account-ARN:key/key_id",
    "CreationDate": "time",
    "Enabled": true,
    "Description": "",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "KeyState": "Enabled",
    "Origin": "AWS_KMS",
    "KeyManager": "CUSTOMER",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ],
    "MultiRegion": false
  }
}
```

- Erstellen Sie mithilfe eines Texteditors eine Datei mit dem Namen `bucketencryption.json` mit folgenden Inhalten.

```
{
  "Rules": [
    {
      "ApplyServerSideEncryptionByDefault": {
        "SSEAlgorithm": "aws:kms",
        "KMSMasterKeyID": "{KMS Key ARN}"
      },
      "BucketKeyEnabled": true
    }
  ]
}
```

- Geben Sie den folgenden Befehl ein und *amzn-s3-demo-bucket* ersetzen Sie ihn durch den Namen des Buckets, in den Sie Protokolle exportieren.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-configuration file://bucketencryption.json
```

Wenn der Befehl keinen Fehler zurückgibt, ist der Vorgang erfolgreich abgeschlossen.

Erstellen Sie eine Exportaufgabe (CLI)

Verwenden Sie den folgenden Befehl, um die Exportaufgabe zu erstellen. Nach der Erstellung kann der Exportvorgang von einigen Sekunden bis zu einigen Stunden dauern. Dies ist abhängig von der Größe der zu exportierenden Daten.

Um Daten nach Amazon S3 zu exportieren, verwenden Sie AWS CLI

1. Melden Sie sich mit ausreichenden Berechtigungen an, wie unter [Zugriffsberechtigungen einrichten \(CLI\)](#) dokumentiert.
2. Verwenden Sie in einer Befehlszeile den folgenden [create-export-task](#) Befehl, um die Exportaufgabe zu erstellen.

```
aws logs create-export-task --profile CWLExportUser --task-name "my-log-group-09-10-2015" --log-group-name "my-log-group" --from 1441490400000 --to 1441494000000 --destination "amzn-s3-demo-bucket" --destination-prefix "export-task-output"
```

Es folgt eine Beispielausgabe.

```
{
  "taskId": "cda45419-90ea-4db5-9833-aade86253e66"
}
```

Kontoübergreifender Export (CLI)

Wenn sich der Amazon-S3-Bucket in einem anderen Konto befindet als die zu exportierenden Protokolle, verwenden Sie die Anweisungen in diesem Abschnitt.

Themen

- [Erstellen Sie einen S3-Bucket für den kontoübergreifenden Export \(CLI\)](#)

- [Zugriffsberechtigungen für den kontoübergreifenden Export \(CLI\) einrichten](#)
- [Berechtigungen für einen S3-Bucket für den kontoübergreifenden Export \(CLI\) festlegen](#)
- [\(Optional\) Exportieren in einen Amazon S3 S3-Ziel-Bucket, der mit SSE-KMS für den kontoübergreifenden Export \(CLI\) verschlüsselt ist](#)
- [Erstellen Sie eine Exportaufgabe für den kontoübergreifenden Export \(CLI\)](#)

Erstellen Sie einen S3-Bucket für den kontoübergreifenden Export (CLI)

Wir empfehlen, dass Sie einen Bucket verwenden, der speziell für CloudWatch Logs erstellt wurde. Wenn Sie jedoch einen vorhandenen Bucket verwenden möchten, gehen Sie direkt zu Schritt 2.

Note

Der S3-Bucket muss sich in derselben Region wie die Protokolldaten befinden, die exportiert werden sollen. CloudWatch Logs unterstützt den Export von Daten in S3-Buckets in einer anderen Region nicht.

Um einen S3-Bucket mit dem zu erstellen AWS CLI

Führen Sie an der Eingabeaufforderung den Befehl [create-bucket](#) aus, wobei LocationConstraint die Region ist, in der Sie Protokolldaten exportieren.

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --create-bucket-configuration  
LocationConstraint=us-east-2
```

Es folgt eine Beispielausgabe.

```
{  
  "Location": "/amzn-s3-demo-bucket"  
}
```

Zugriffsberechtigungen für den kontoübergreifenden Export (CLI) einrichten

Zunächst müssen Sie eine neue IAM-Richtlinie erstellen, damit CloudWatch Logs die `s3:PutObject` Aktion für den Amazon S3 S3-Ziel-Bucket im Zielkonto hat.

Neben der `s3:PutObject` Aktion hängen weitere in der Richtlinie enthaltene Aktionen davon ab, ob der Ziel-Bucket AWS KMS Verschlüsselung verwendet oder mit der Einstellung [S3-Objektbesitz ACLs](#) aktiviert wurde.

- Wenn Sie die KMS-Verschlüsselung verwenden, fügen Sie die `kms:Decrypt` Aktionen `kms:GenerateDataKey` und für die Schlüsselressource hinzu
- Wenn sie im Bucket aktiviert ACLs sind, fügen Sie die `s3:PutObjectAcl` Aktion für die Bucket-Ressource hinzu

Ändern `amzn-s3-demo-bucket` Sie in den folgenden Richtlinien den Namen Ihres Ziel-S3-Buckets.

Die Richtlinie, die Sie erstellen, hängt davon ab, ob der Ziel-Bucket AWS KMS -Verschlüsselung verwendet. Wenn es keine AWS KMS Verschlüsselung verwendet, erstellen Sie eine Richtlinie mit dem folgenden Inhalt.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
  ]
}
```

Wenn der Ziel-Bucket AWS KMS Verschlüsselung verwendet, erstellen Sie eine Richtlinie mit dem folgenden Inhalt.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "s3:PutObject",
```

```

        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
        ],
        "Resource": "arn:aws:kms:us-east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
]
}

```

Wenn sie im Ziel-Bucket aktiviert ACLs sind, fügen Sie in den obigen Richtlinien den Block s3: PutObjectAcl zum Block s3: PutObject Action hinzu.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
  ]
}

```

Um eine Exportaufgabe zu erstellen, müssen Sie mit einer IAM-Rolle angemeldet sein, der die AmazonS3ReadOnlyAccess verwaltete Richtlinie und die oben erstellte IAM-Richtlinie zugeordnet sind. Außerdem müssen Sie über die folgenden Berechtigungen verfügen:

- logs:CreateExportTask
- logs:CancelExportTask

- `logs:DescribeExportTasks`
- `logs:DescribeLogStreams`
- `logs:DescribeLogGroups`

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in: AWS IAM Identity Center

Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter [Erstellen eines Berechtigungssatzes](#) im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter [Eine Rolle für einen externen Identitätsanbieter \(Verbund\) erstellen](#) im IAM-Benutzerhandbuch.

- IAM-Benutzer:
 - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter [Eine Rolle für einen IAM-Benutzer erstellen](#) im IAM-Benutzerhandbuch.
 - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter [Hinzufügen von Berechtigungen zu einem Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

Berechtigungen für einen S3-Bucket für den kontoübergreifenden Export (CLI) festlegen

Standardmäßig werden alle S3-Buckets und -Objekte als privat eingestuft. Nur der Ressourceneigentümer, das Konto, in dem der Bucket erstellt wurde, kann auf den Bucket und alle darin enthaltenen Objekte zugreifen. Der Ressourcenbesitzer kann jedoch anderen Ressourcen und Benutzern Zugriffsberechtigungen gewähren, indem er eine Zugriffsrichtlinie schreibt.

Important

Um Exporte in S3-Buckets sicherer zu machen, müssen Sie jetzt die Liste der Quellkonten angeben, die Protokolldaten in Ihren S3-Bucket exportieren dürfen.

Im folgenden Beispiel entspricht die Liste der Konten IDs im `aws:SourceAccount` Schlüssel den Konten, von denen ein Benutzer Protokolldaten in Ihren S3-Bucket exportieren kann. Der Schlüssel `aws:SourceArn` ist die Ressource, für die die Aktion ausgeführt wird. Sie können

dies auf eine bestimmte Protokollgruppe beschränken oder einen Platzhalter verwenden, wie in diesem Beispiel zu sehen.

Wir empfehlen, dass Sie auch die Konto-ID des Kontos angeben, in dem der S3-Bucket erstellt wurde, um den Export innerhalb desselben Kontos zu ermöglichen.

So legen Sie Berechtigungen für einen S3-Bucket fest

1. Erstellen Sie eine Datei mit dem Namen `policy.json` und fügen Sie dem Endpunkt der `amzn-s3-demo-bucket` Region, in die Sie Protokolldaten exportieren, die folgende Zugriffsrichtlinie `Principal` hinzu, wobei Sie den Namen Ihres Ziel-S3-Buckets annehmen, z. `us-west-1` B. Verwenden Sie einen Text-Editor, um diese Richtliniendatei zu erstellen. Verwenden Sie nicht die IAM-Konsole.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "s3:GetBucketAcl",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": [
            "123456789012",
            "111122223333"
          ]
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:logs:us-east-1:123456789012:log-group:*",
            "arn:aws:logs:us-east-1:111122223333:log-group:*"
          ]
        }
      }
    }
  ],
  {
```

```

    "Action": "s3:PutObject" ,
    "Effect": "Allow",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
    "Condition": {
      "StringEquals": {
        "s3:x-amz-acl": "bucket-owner-full-control",
        "aws:SourceAccount": [
          "123456789012",
          "111122223333"
        ]
      },
      "ArnLike": {
        "aws:SourceArn": [
          "arn:aws:logs:us-east-1:123456789012:log-group:*",
          "arn:aws:logs:us-east-1:111122223333:log-group:*"
        ]
      }
    }
  },
  {
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::111122223333:role/role_name"
    },
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::>amzn-s3-demo-bucket/*",
    "Condition": {
      "StringEquals": {
        "s3:x-amz-acl": "bucket-owner-full-control"
      }
    }
  }
]
}

```

2. Legen Sie mithilfe des [put-bucket-policy](#) Befehls die Richtlinie, die Sie gerade hinzugefügt haben, als Zugriffsrichtlinie für Ihren Bucket fest. Diese Richtlinie ermöglicht CloudWatch Logs, Protokolldaten in Ihren S3-Bucket zu exportieren. Der Bucket-Eigentümer hat vollen Zugriff auf alle exportierten Objekte.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://policy.json
```

⚠ Warning

Wenn mit dem vorhandenen Bucket bereits eine oder mehrere Richtlinien verknüpft sind, fügen Sie die Anweisungen für den Zugriff auf CloudWatch Logs zu dieser Richtlinie oder diesen Richtlinien hinzu. Sie sollten eine Beurteilung der daraus resultierenden Berechtigungen vornehmen, um sicherzustellen, dass sie für die Benutzer, die auf den Bucket zugreifen werden, geeignet sind.

(Optional) Exportieren in einen Amazon S3 S3-Ziel-Bucket, der mit SSE-KMS für den kontoübergreifenden Export (CLI) verschlüsselt ist

Dieses Verfahren ist nur erforderlich, wenn Sie in einen S3-Bucket exportieren, der serverseitige Verschlüsselung mit verwendet. AWS KMS keys Diese Verschlüsselung wird als SSE-KMS bezeichnet.

So exportieren Sie in einen mit SSE-KMS verschlüsselten Bucket

1. Verwenden Sie einen Texteditor, um eine Datei mit dem Namen `key_policy.json` zu erstellen und die folgende Zugriffsrichtlinie hinzuzufügen. Wenn Sie die Richtlinie hinzufügen, nehmen Sie die folgenden Änderungen vor:
 - `us-east-1` Ersetzen Sie es durch die Region Ihrer Protokolle.
 - `account-ARN` Ersetzen Sie ihn durch den ARN des Kontos, dem der KMS-Schlüssel gehört.
 - `123456789012` Ersetzen Sie ihn durch die Kontonummer, der den KMS-Schlüssel besitzt.
 - `key_id` mit der KMS-Schlüssel-ID.
 - `role_name` mit der Rolle, die für die Erstellung der Exportaufgabe verwendet wurde.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCWServicePrincipalUsage",
```

```

    "Effect": "Allow",
    "Principal": {
      "Service": "logs.us-east-1.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey",
      "kms:Decrypt"
    ],
    "Resource": "*"
  },
  {
    "Sid": "EnableIAMUserPermissions",
    "Effect": "Allow",
    "Principal": {
      "AWS": "account-ARN"
    },
    "Action": [
      "kms:GetKeyPolicy*",
      "kms:PutKeyPolicy*",
      "kms:DescribeKey*",
      "kms:CreateAlias*",
      "kms:ScheduleKeyDeletion*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  },
  {
    "Sid": "EnableIAMRolePermissions",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::111122223333:role/role_name"
    },
    "Action": [
      "kms:GenerateDataKey",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:us-east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }
]
}

```

2. Geben Sie den folgenden Befehl ein:

```
aws kms create-key --policy file:///key_policy.json
```

Das Folgende ist eine Beispielausgabe aus diesem Befehl:

```
{
  "KeyMetadata": {
    "AWSAccountId": "account_id",
    "KeyId": "key_id",
    "Arn": "arn:aws:kms:us-east-1:123456789012:key/key_id",
    "CreationDate": "time",
    "Enabled": true,
    "Description": "",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "KeyState": "Enabled",
    "Origin": "AWS_KMS",
    "KeyManager": "CUSTOMER",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ],
    "MultiRegion": false
  }
}
```

- Erstellen Sie mithilfe eines Texteditors eine Datei mit dem Namen `bucketencryption.json` mit folgenden Inhalten.

```
{
  "Rules": [
    {
      "ApplyServerSideEncryptionByDefault": {
        "SSEAlgorithm": "aws:kms",
        "KMSMasterKeyID": "{KMS Key ARN}"
      },
      "BucketKeyEnabled": true
    }
  ]
}
```

- Geben Sie den folgenden Befehl ein und `amzn-s3-demo-bucket` ersetzen Sie ihn durch den Namen des Buckets, in den Sie Logs exportieren.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-configuration file://bucketencryption.json
```

Wenn der Befehl keinen Fehler zurückgibt, ist der Vorgang erfolgreich abgeschlossen.

Erstellen Sie eine Exportaufgabe für den kontoübergreifenden Export (CLI)

Verwenden Sie den folgenden Befehl, um die Exportaufgabe zu erstellen. Nach der Erstellung kann der Exportvorgang von einigen Sekunden bis zu einigen Stunden dauern. Dies ist abhängig von der Größe der zu exportierenden Daten.

Um Daten nach Amazon S3 zu exportieren, verwenden Sie AWS CLI

1. Melden Sie sich mit ausreichenden Berechtigungen an, wie unter [Zugriffsberechtigungen einrichten \(CLI\)](#) dokumentiert.
2. Verwenden Sie in einer Befehlszeile den folgenden [create-export-task](#) Befehl, um die Exportaufgabe zu erstellen.

```
aws logs create-export-task --profile CWLExportUser --task-name "my-log-group-09-10-2015" --log-group-name "my-log-group" --from 1441490400000 --to 1441494000000 --destination "amzn-s3-demo-bucket" --destination-prefix "export-task-output"
```

Es folgt eine Beispielausgabe.

```
{  
  "taskId": "cda45419-90ea-4db5-9833-aade86253e66"  
}
```

Exportaufgaben beschreiben (CLI)

Nachdem Sie eine Exportaufgabe erstellt haben, können Sie den aktuellen Status der Aufgabe abrufen.

Um Exportaufgaben mit dem zu beschreiben AWS CLI

Verwenden Sie in einer Befehlszeile den folgenden [describe-export-tasks](#) Befehl.

```
aws logs --profile CWLEXPOTUser describe-export-tasks --task-id  
"cda45419-90ea-4db5-9833-aade86253e66"
```

Es folgt eine Beispielausgabe.

```
{  
  "exportTasks": [  
    {  
      "destination": "my-exported-logs",  
      "destinationPrefix": "export-task-output",  
      "executionInfo": {  
        "creationTime": 1441495400000  
      },  
      "from": 1441490400000,  
      "logGroupName": "my-log-group",  
      "status": {  
        "code": "RUNNING",  
        "message": "Started Successfully"  
      },  
      "taskId": "cda45419-90ea-4db5-9833-aade86253e66",  
      "taskName": "my-log-group-09-10-2015",  
      "tTo": 1441494000000  
    }  
  ]  
}
```

Sie können den `describe-export-tasks`-Befehl auf drei verschiedene Arten verwenden:

- Ohne Filter – Listet alle Exportvorgänge in umgekehrter Reihenfolge der Erstellung auf.
- Filtern auf Aufgaben-ID – Listet die Exportaufgabe (falls vorhanden) mit der angegebenen ID auf.
- Filtern auf Aufgabenstatus – Listet die Exportaufgaben mit dem angegebenen Status auf.

Verwenden Sie z. B. den folgenden Befehl, um einen Filter für den Status `FAILED` zu nutzen.

```
aws logs --profile CWLEXPOTUser describe-export-tasks --status-code "FAILED"
```

Es folgt eine Beispielausgabe.

```
{  
  "exportTasks": [  
    {
```



```
{
  "destination": "amzn-s3-demo-bucket",
  "destinationPrefix": "export-task-output",
  "executionInfo": {
    "completionTime": 1441498600000
    "creationTime": 1441495400000
  },
  "from": 1441490400000,
  "logGroupName": "my-log-group",
  "status": {
    "code": "FAILED",
    "message": "FAILED"
  },
  "taskId": "cda45419-90ea-4db5-9833-aade86253e66",
  "taskName": "my-log-group-09-10-2015",
  "to": 1441494000000
}]
}
```

Eine Exportaufgabe abbrechen (CLI)

Sie können eine Exportaufgabe stornieren, wenn sie entweder einen Status PENDING oder RUNNING aufweist.

Um eine Exportaufgabe mit dem abzuberechnen AWS CLI

Verwenden Sie in einer Befehlszeile den folgenden [cancel-export-task](#) Befehl:

```
aws logs --profile CWExportUser cancel-export-task --task-id "cda45419-90ea-4db5-9833-aade86253e66"
```

Sie können den [describe-export-tasks](#) Befehl verwenden, um zu überprüfen, ob die Aufgabe erfolgreich abgebrochen wurde.

Streaming CloudWatch protokolliert Daten an Amazon OpenSearch Service

Sie können eine Protokollgruppe in Amazon CloudWatch Logs konfigurieren, sodass Sie Daten nahezu in Echtzeit in Ihren Amazon OpenSearch Service-Cluster streamen können. Weitere Informationen finden Sie unter [Echtzeitverarbeitung von Protokolldaten mit Abonnements](#).

Note

Streaming to OpenSearch Service wird nur für Protokollgruppen der Standard-Protokollklasse unterstützt. Weitere Hinweise zu Protokollklassen finden Sie unter [Klassen protokollieren](#).

Abhängig von der Menge der gestreamten Protokolldaten sollten Sie erwägen, ein Parallelitätslimit auf Funktionsebene festzulegen. Weitere Informationen erhalten Sie unter [Skalierung von Lambda-Funktionen](#).

Note

Da das Streamen großer Mengen von CloudWatch Logs-Daten an den OpenSearch Service zu hohen Nutzungsgebühren führen kann, empfehlen wir, dass Sie in der Konsole ein Budget erstellen. AWS Fakturierung und Kostenmanagement Weitere Informationen finden Sie unter [Kosten mit AWS Budgets verwalten](#).

In diesem Abschnitt werden die Voraussetzungen beschrieben, die Sie erfüllen müssen, bevor Sie OpenSearch Service für eine Protokollgruppe abonnieren können. Außerdem wird beschrieben, wie Sie eine Protokollgruppe für OpenSearch Service abonnieren.

Voraussetzungen

Bevor Sie beginnen, erstellen Sie eine OpenSearch Dienstdomäne. Die Domäne kann entweder öffentlichen Zugriff oder VPC-Zugriff haben, aber Sie können den Zugriffstyp nicht ändern, nachdem die Domäne erstellt wurde. Möglicherweise möchten Sie Ihre OpenSearch Service-Domain-Einstellungen später überprüfen und Ihre Clusterkonfiguration entsprechend der Datenmenge, die

Ihr Cluster verarbeiten wird, ändern. Anweisungen zum Erstellen einer Domäne finden Sie unter [OpenSearch Dienstdomänen erstellen](#).

Weitere Informationen zu OpenSearch Service finden Sie im [Amazon OpenSearch Service Developer Guide](#).

Abonnieren Sie OpenSearch Service für eine Protokollgruppe

Sie können die CloudWatch Konsole verwenden, um eine Protokollgruppe für OpenSearch Service zu abonnieren.

Um eine Protokollgruppe für OpenSearch Service zu abonnieren

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Protokollgruppen aus.
3. Wählen Sie den Namen der Protokollgruppe aus.
4. Wählen Sie Aktionen, Abonnementfilter, Amazon OpenSearch Service-Abonnementfilter erstellen.
5. Wählen Sie aus, ob Sie zu einem Cluster in diesem Konto oder einem anderen Konto streamen möchten.
 - Wählen Sie bei Auswahl dieses Kontos die Domäne aus, die Sie im vorherigen Schritt erstellt haben.
 - Wenn Sie ein anderes Konto gewählt haben, geben Sie den Domänen-ARN und den Endpunkt an.
6. Wählen Sie für Lambda IAM Execution Role die IAM-Rolle aus, die Lambda bei der Ausführung von Aufrufen verwenden soll. OpenSearch

Die IAM-Rolle, die Sie auswählen, muss diese Anforderungen erfüllen:

- Es muss `lambda.amazonaws.com` im Verhältnis stehen.
- Die Richtlinie muss Folgendes umfassen:

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "logs:CreateLogGroup",  
      "Resource": "arn:aws:logs:*:*:log-group:/*",  
      "Principal": "lambda.amazonaws.com",  
      "Condition": {  
        "StringEquals": {  
          "aws:PrincipalArn": "arn:aws:lambda:*:*:function:*" }  
      }  
    }  
  ]  
}
```

```
{
  "Sid": "AllowOpenSearchStreamingAccess",
  "Action": [
    "es:*"
  ],
  "Effect": "Allow",
  "Resource": "arn:aws:es:us-  
east-1:123456789012:domain/cloudwatch-logs/*"
}
```

- Wenn die OpenSearch Zieldienstdomäne VPC-Zugriff verwendet, muss die AWSLambdaVPCLambdaAccessExecutionRoleRichtlinie an die Rolle angehängt sein. Diese von Amazon verwaltete Richtlinie gewährt Lambda Zugriff auf die VPC des Kunden, sodass Lambda auf den Endpunkt in der OpenSearch VPC schreiben kann.
7. Wählen Sie unter Log Format (Protokollformat) ein Protokollformat aus.
 8. Geben Sie unter Subscription Filter Pattern (Abonnementfiltermuster) die Begriffe oder Muster ein, die in Ihren Protokollereignissen zu finden sind. Dadurch wird sichergestellt, dass Sie nur die Daten, an denen Sie interessiert sind, an Ihren Cluster senden. OpenSearch Weitere Informationen finden Sie unter [Erstellen von Metriken aus Protokollereignissen mithilfe von Filtern](#).
 9. (Optional) Wählen Sie unter Select log data to test (Zu testende Protokolldaten auswählen) einen Protokoll-Stream und anschließend Test pattern (Muster testen) aus, um zu überprüfen, ob Ihr Suchfilter die erwarteten Ergebnisse zurückgeben.
 10. Wählen Sie dann Start Streaming (Streamen starten).

Codebeispiele für CloudWatch Logs mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie CloudWatch Logs mit einem AWS Software Development Kit (SDK) verwendet wird.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Codebeispiele

- [Grundlegende Beispiele für CloudWatch Logs mit AWS SDKs](#)
 - [Aktionen für CloudWatch Logs mit AWS SDKs](#)
 - [AssociateKmsKeyMit einem AWS SDK verwenden](#)
 - [CancelExportTaskMit einem AWS SDK verwenden](#)
 - [CreateExportTaskMit einem AWS SDK verwenden](#)
 - [Verwendung CreateLogGroup mit einem AWS SDK oder CLI](#)
 - [Verwendung CreateLogStream mit einem AWS SDK oder CLI](#)
 - [Verwendung DeleteLogGroup mit einem AWS SDK oder CLI](#)
 - [DeleteSubscriptionFilterMit einem AWS SDK verwenden](#)
 - [DescribeExportTasksMit einem AWS SDK verwenden](#)
 - [Verwendung DescribeLogGroups mit einem AWS SDK oder CLI](#)
 - [Verwendung DescribeLogStreams mit einem AWS SDK oder CLI](#)
 - [DescribeSubscriptionFiltersMit einem AWS SDK verwenden](#)
 - [Verwendung GetLogEvents mit einem AWS SDK oder CLI](#)
 - [GetQueryResultsMit einem AWS SDK verwenden](#)
 - [PutSubscriptionFilterMit einem AWS SDK verwenden](#)
 - [StartLiveTailMit einem AWS SDK verwenden](#)

- [StartQueryMit einem AWS SDK verwenden](#)
- [Szenarien für CloudWatch Logs mit AWS SDKs](#)
 - [Amazon ECS Service Connect konfigurieren](#)
 - [Erstellen Sie Ihre erste Lambda-Funktion](#)
 - [Verwenden Sie CloudWatch Logs, um eine umfangreiche Abfrage auszuführen](#)
 - [Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion](#)

Grundlegende Beispiele für CloudWatch Logs mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon CloudWatch Logs mit verwenden können AWS SDKs.

Beispiele

- [Aktionen für CloudWatch Logs mit AWS SDKs](#)
 - [AssociateKmsKeyMit einem AWS SDK verwenden](#)
 - [CancelExportTaskMit einem AWS SDK verwenden](#)
 - [CreateExportTaskMit einem AWS SDK verwenden](#)
 - [Verwendung CreateLogGroup mit einem AWS SDK oder CLI](#)
 - [Verwendung CreateLogStream mit einem AWS SDK oder CLI](#)
 - [Verwendung DeleteLogGroup mit einem AWS SDK oder CLI](#)
 - [DeleteSubscriptionFilterMit einem AWS SDK verwenden](#)
 - [DescribeExportTasksMit einem AWS SDK verwenden](#)
 - [Verwendung DescribeLogGroups mit einem AWS SDK oder CLI](#)
 - [Verwendung DescribeLogStreams mit einem AWS SDK oder CLI](#)
 - [DescribeSubscriptionFiltersMit einem AWS SDK verwenden](#)
 - [Verwendung GetLogEvents mit einem AWS SDK oder CLI](#)
 - [GetQueryResultsMit einem AWS SDK verwenden](#)
 - [PutSubscriptionFilterMit einem AWS SDK verwenden](#)
 - [StartLiveTailMit einem AWS SDK verwenden](#)
 - [StartQueryMit einem AWS SDK verwenden](#)

Aktionen für CloudWatch Logs mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne CloudWatch Logs-Aktionen mit ausgeführt AWS SDKs werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die CloudWatch Logs-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter [Szenarien für CloudWatch Logs mit AWS SDKs](#) anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der [Amazon CloudWatch Logs API-Referenz](#).

Beispiele

- [AssociateKmsKeyMit einem AWS SDK verwenden](#)
- [CancelExportTaskMit einem AWS SDK verwenden](#)
- [CreateExportTaskMit einem AWS SDK verwenden](#)
- [Verwendung CreateLogGroup mit einem AWS SDK oder CLI](#)
- [Verwendung CreateLogStream mit einem AWS SDK oder CLI](#)
- [Verwendung DeleteLogGroup mit einem AWS SDK oder CLI](#)
- [DeleteSubscriptionFilterMit einem AWS SDK verwenden](#)
- [DescribeExportTasksMit einem AWS SDK verwenden](#)
- [Verwendung DescribeLogGroups mit einem AWS SDK oder CLI](#)
- [Verwendung DescribeLogStreams mit einem AWS SDK oder CLI](#)
- [DescribeSubscriptionFiltersMit einem AWS SDK verwenden](#)
- [Verwendung GetLogEvents mit einem AWS SDK oder CLI](#)
- [GetQueryResultsMit einem AWS SDK verwenden](#)
- [PutSubscriptionFilterMit einem AWS SDK verwenden](#)
- [StartLiveTailMit einem AWS SDK verwenden](#)
- [StartQueryMit einem AWS SDK verwenden](#)

AssociateKmsKeyMit einem AWS SDK verwenden

Das folgende Codebeispiel zeigt, wie es verwendet wirdAssociateKmsKey.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to associate an AWS Key Management Service (AWS KMS) key with
/// an Amazon CloudWatch Logs log group.
/// </summary>
public class AssociateKmsKey
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        string kmsKeyId = "arn:aws:kms:us-west-2:<account-
number>:key/7c9ecce2-38cb-4c4f-9db3-766ee8dd3ad4";
        string groupName = "cloudwatchlogs-example-loggroup";

        var request = new AssociateKmsKeyRequest
        {
            KmsKeyId = kmsKeyId,
            LogGroupName = groupName,
        };

        var response = await client.AssociateKmsKeyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```



```

        {
            Console.WriteLine($"Successfully associated KMS key ID:
{kmsKeyId} with log group: {groupName}.");
        }
        else
        {
            Console.WriteLine("Could not make the association between:
{kmsKeyId} and {groupName}.");
        }
    }
}

```

- Einzelheiten zur API finden Sie [AssociateKmsKey](#) in der AWS SDK für .NET API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

CancelExportTask Mit einem AWS SDK verwenden

Das folgende Codebeispiel zeigt, wie es verwendet wird `CancelExportTask`.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to cancel an Amazon CloudWatch Logs export task.

```

```
/// </summary>
public class CancelExportTask
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string taskId = "exampleTaskId";

        var request = new CancelExportTaskRequest
        {
            TaskId = taskId,
        };

        var response = await client.CancelExportTaskAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"{taskId} successfully canceled.");
        }
        else
        {
            Console.WriteLine($"{taskId} could not be canceled.");
        }
    }
}
```

- Einzelheiten zur API finden Sie [CancelExportTask](#) in der AWS SDK für .NET API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

CreateExportTask Mit einem AWS SDK verwenden

Das folgende Codebeispiel zeigt, wie es verwendet wird `CreateExportTask`.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Export Task to export the contents of the Amazon
/// CloudWatch Logs to the specified Amazon Simple Storage Service (Amazon
S3)
/// bucket.
/// </summary>
public class CreateExportTask
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string taskName = "export-task-example";
        string logGroupName = "cloudwatchlogs-example-loggroup";
        string destination = "amzn-s3-demo-bucket";
        var fromTime = 1437584472382;
        var toTime = 1437584472833;

        var request = new CreateExportTaskRequest
        {
            From = fromTime,
            To = toTime,
            TaskName = taskName,
            LogGroupName = logGroupName,
```

```
        Destination = destination,
    };

    var response = await client.CreateExportTaskAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"The task, {taskName} with ID: " +
                           $"{response.TaskId} has been created
successfully.");
    }
}
}
```

- Einzelheiten zur API finden Sie [CreateExportTask](#) in der AWS SDK für .NET API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **CreateLogGroup** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie CreateLogGroup verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Amazon ECS Service Connect konfigurieren](#)

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Amazon CloudWatch Logs log group.
/// </summary>
public class CreateLogGroup
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        string logGroupName = "cloudwatchlogs-example-loggroup";

        var request = new CreateLogGroupRequest
        {
            LogGroupName = logGroupName,
        };

        var response = await client.CreateLogGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully create log group with ID:
{logGroupName}.");
        }
        else
        {
            Console.WriteLine("Could not create log group.");
        }
    }
}
```

- Einzelheiten zur API finden Sie [CreateLogGroup](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl erstellt eine Protokollgruppe mit dem Namen `my-logs`.

```
aws logs create-log-group --log-group-name my-logs
```

- Einzelheiten zur API finden Sie [CreateLogGroup](#) in der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
import { CreateLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new CreateLogGroupCommand({
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Einzelheiten zur API finden Sie [CreateLogGroup](#) in der AWS SDK für JavaScript API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **CreateLogStream** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie CreateLogStream verwendet wird.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Amazon CloudWatch Logs stream for a CloudWatch
/// log group.
/// </summary>
public class CreateLogStream
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string logGroupName = "cloudwatchlogs-example-loggroup";
        string logStreamName = "cloudwatchlogs-example-logstream";

        var request = new CreateLogStreamRequest
        {
            LogGroupName = logGroupName,
```

```
        LogStreamName = logStreamName,
    };

    var response = await client.CreateLogStreamAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{logStreamName} successfully created for {logGroupName}.");
    }
    else
    {
        Console.WriteLine("Could not create stream.");
    }
}
}
```

- Einzelheiten zur API finden Sie [CreateLogStream](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Im folgenden Beispiel wird ein Protokollstream mit dem Namen `20150601` in der Protokollgruppe `my-logs` erstellt.

```
aws logs create-log-stream --log-group-name my-logs --log-stream-name 20150601
```

- Einzelheiten zur API finden Sie [CreateLogStream](#) in der AWS CLI Befehlsreferenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **DeleteLogGroup** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteLogGroup` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Amazon ECS Service Connect konfigurieren](#)
- [Erstellen Sie Ihre erste Lambda-Funktion](#)

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Uses the Amazon CloudWatch Logs Service to delete an existing
/// CloudWatch Logs log group.
/// </summary>
public class DeleteLogGroup
{
    public static async Task Main()
    {
        var client = new AmazonCloudWatchLogsClient();
        string logGroupName = "cloudwatchlogs-example-loggroup";

        var request = new DeleteLogGroupRequest
        {
            LogGroupName = logGroupName,
        };

        var response = await client.DeleteLogGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {

```

```
        Console.WriteLine($"Successfully deleted CloudWatch log group,  
        {logGroupName}.");  
    }  
}
```

- Einzelheiten zur API finden Sie [DeleteLogGroup](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird eine Protokollgruppe mit dem Namen my-logs gelöscht:

```
aws logs delete-log-group --log-group-name my-logs
```

- Einzelheiten zur API finden Sie [DeleteLogGroup](#) in der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
import { DeleteLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";  
import { client } from "../libs/client.js";  
  
const run = async () => {  
    const command = new DeleteLogGroupCommand({  
        // The name of the log group.  
        logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,  
    });  
  
    try {
```

```
    return await client.send(command);  
  } catch (err) {  
    console.error(err);  
  }  
};  
  
export default run();
```

- Einzelheiten zur API finden Sie [DeleteLogGroup](#) in der AWS SDK für JavaScript API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

DeleteSubscriptionFilter Mit einem AWS SDK verwenden

Die folgenden Code-Beispiele zeigen, wie DeleteSubscriptionFilter verwendet wird.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>  
#include <aws/core/utils/Outcome.h>  
#include <aws/logs/CloudWatchLogsClient.h>  
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>  
#include <iostream>
```

Löschen Sie den Abonnementfilter.

```

Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::DeleteSubscriptionFilterRequest request;
request.SetFilterName(filter_name);
request.SetLogGroupName(log_group);

auto outcome = cwl.DeleteSubscriptionFilter(request);
if (!outcome.IsSuccess()) {
    std::cout << "Failed to delete CloudWatch log subscription filter "
        << filter_name << ": " << outcome.GetError().GetMessage() <<
        std::endl;
} else {
    std::cout << "Successfully deleted CloudWatch logs subscription " <<
        "filter " << filter_name << std::endl;
}

```

- Einzelheiten zur API finden Sie [DeleteSubscriptionFilter](#) in der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */

```

```

* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DeleteSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <filter> <logGroup>

            Where:
                filter - The name of the subscription filter (for example,
MyFilter).
                logGroup - The name of the log group. (for example, testgroup).
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String logGroup = args[1];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
            .build();

        deleteSubFilter(logs, filter, logGroup);
        logs.close();
    }

    public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
String logGroup) {
        try {
            DeleteSubscriptionFilterRequest request =
DeleteSubscriptionFilterRequest.builder()
                .filterName(filter)
                .logGroupName(logGroup)
                .build();

            logs.deleteSubscriptionFilter(request);
            System.out.printf("Successfully deleted CloudWatch logs subscription
filter %s", filter);

        } catch (CloudWatchException e) {

```

```

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Einzelheiten zur API finden Sie [DeleteSubscriptionFilter](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

import { DeleteSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteSubscriptionFilterCommand({
    // The name of the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();

```

- Einzelheiten zur API finden Sie [DeleteSubscriptionFilter](#) in der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cw1 = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  filterName: "FILTER",
  logGroupName: "LOG_GROUP",
};

cw1.deleteSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [DeleteSubscriptionFilter](#) in der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
suspend fun deleteSubFilter(
    filter: String?,
    logGroup: String?,
) {
    val request =
        DeleteSubscriptionFilterRequest {
            filterName = filter
            logGroupName = logGroup
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
$filter")
    }
}
```

- Einzelheiten zur API finden Sie [DeleteSubscriptionFilter](#) in der API-Referenz zum AWS SDK für Kotlin.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

DescribeExportTasks Mit einem AWS SDK verwenden

Das folgende Codebeispiel zeigt, wie es verwendet wird `DescribeExportTasks`.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to retrieve a list of information about Amazon CloudWatch
/// Logs export tasks.
/// </summary>
public class DescribeExportTasks
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        var request = new DescribeExportTasksRequest
        {
            Limit = 5,
        };

        var response = new DescribeExportTasksResponse();

        do
        {
            response = await client.DescribeExportTasksAsync(request);
            response.ExportTasks.ForEach(t =>
            {
```

```
        Console.WriteLine($"{t.TaskName} with ID: {t.TaskId} has  
status: {t.Status}");  
    });  
}  
while (response.NextToken is not null);  
}  
}
```

- Einzelheiten zur API finden Sie [DescribeExportTasks](#) in der AWS SDK für .NET API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **DescribeLogGroups** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie DescribeLogGroups verwendet wird.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
using System;  
using System.Threading.Tasks;  
using Amazon.CloudWatchLogs;  
using Amazon.CloudWatchLogs.Model;  
  
/// <summary>  
/// Retrieves information about existing Amazon CloudWatch Logs log groups  
/// and displays the information on the console.  
/// </summary>  
public class DescribeLogGroups
```

```
{
    public static async Task Main()
    {
        // Creates a CloudWatch Logs client using the default
        // user. If you need to work with resources in another
        // AWS Region than the one defined for the default user,
        // pass the AWS Region as a parameter to the client constructor.
        var client = new AmazonCloudWatchLogsClient();

        bool done = false;
        string newToken = null;

        var request = new DescribeLogGroupsRequest
        {
            Limit = 5,
        };

        DescribeLogGroupsResponse response;

        do
        {
            if (newToken is not null)
            {
                request.NextToken = newToken;
            }

            response = await client.DescribeLogGroupsAsync(request);

            response.LogGroups.ForEach(lg =>
            {
                Console.WriteLine($"{lg.LogGroupName} is associated with the
key: {lg.KmsKeyId}.");
                Console.WriteLine($"Created on:
{lg.CreationTime.Date.Date}");
                Console.WriteLine($"Date for this group will be stored for:
{lg.RetentionInDays} days.\n");
            });

            if (response.NextToken is null)
            {
                done = true;
            }
            else
            {

```

```
        newToken = response.NextToken;
    }
}
while (!done);
}
}
```

- Einzelheiten zur API finden Sie [DescribeLogGroups](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl beschreibt eine Protokollgruppe mit dem Namen my-logs:

```
aws logs describe-log-groups --log-group-name-prefix my-logs
```

Ausgabe:

```
{
  "logGroups": [
    {
      "storedBytes": 0,
      "metricFilterCount": 0,
      "creationTime": 1433189500783,
      "logGroupName": "my-logs",
      "retentionInDays": 5,
      "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:*"
    }
  ]
}
```

- Einzelheiten zur API finden Sie [DescribeLogGroups](#) in der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
import {
  paginateDescribeLogGroups,
  CloudWatchLogsClient,
} from "@aws-sdk/client-cloudwatch-logs";

const client = new CloudWatchLogsClient({});

export const main = async () => {
  const paginatedLogGroups = paginateDescribeLogGroups({ client }, {});
  const logGroups = [];

  for await (const page of paginatedLogGroups) {
    if (page.logGroups?.every((lg) => !!lg)) {
      logGroups.push(...page.logGroups);
    }
  }

  console.log(logGroups);
  return logGroups;
};
```

- Einzelheiten zur API finden Sie [DescribeLogGroups](#) in der AWS SDK für JavaScript API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **DescribeLogStreams** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLogStreams` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erstellen Sie Ihre erste Lambda-Funktion](#)

CLI

AWS CLI

Der folgende Befehl zeigt alle Protokollstreams in der Protokollgruppe `my-logs` an, die mit dem Präfix `2015` beginnen:

```
aws logs describe-log-streams --log-group-name my-logs --log-stream-name-  
prefix 2015
```

Ausgabe:

```
{  
  "logStreams": [  
    {  
      "creationTime": 1433189871774,  
      "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-  
stream:20150531",  
      "logStreamName": "20150531",  
      "storedBytes": 0  
    },  
    {  
      "creationTime": 1433189873898,  
      "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-  
stream:20150601",  
      "logStreamName": "20150601",  
      "storedBytes": 0  
    }  
  ]  
}
```

- Einzelheiten zur API finden Sie [DescribeLogStreams](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Sucht innerhalb einer angegebenen Protokollgruppe nach Protokollströmen, die einem bestimmten Präfix entsprechen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class CloudWatchLogsSearch {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <logGroupName> <logStreamName>

            Where:
                logGroupName - The name of the log group (for example,
                WeathertopJavaContainerLogs).
                logStreamName - The name of the log stream (for example,
                weathertop-java-stream).
                pattern - the pattern to use (for example, INFO)

            "";

        if (args.length != 3) {
            System.out.print(usage);
        }
    }
}
```

```

        System.exit(1);
    }

    String logGroupName = args[0] ;
    String logStreamName = args[1] ;
    String pattern = args[2] ;

    CloudWatchLogsClient cwlClient = CloudWatchLogsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    searchLogStreamsAndFilterEvents(cwlClient, logGroupName, logStreamName,
pattern);
    }

    /**
     * Searches for log streams with a specific prefix within a log group and
     filters log events based on a specified pattern.
     *
     * @param cwlClient      the CloudWatchLogsClient used to interact with AWS
     CloudWatch Logs
     * @param logGroupName  the name of the log group to search within
     * @param logStreamPrefix the prefix of the log streams to search for
     * @param pattern        the pattern to filter log events by
     */
    public static void searchLogStreamsAndFilterEvents(CloudWatchLogsClient
cwlClient, String logGroupName, String logStreamPrefix, String pattern) {
        DescribeLogStreamsRequest describeLogStreamsRequest =
DescribeLogStreamsRequest.builder()
            .logGroupName(logGroupName)
            .logStreamNamePrefix(logStreamPrefix)
            .build();

        DescribeLogStreamsResponse describeLogStreamsResponse =
cwlClient.describeLogStreams(describeLogStreamsRequest);
        List<LogStream> logStreams = describeLogStreamsResponse.logStreams();

        for (LogStream logStream : logStreams) {
            String logStreamName = logStream.logStreamName();
            System.out.println("Searching in log stream: " + logStreamName);

            FilterLogEventsRequest filterLogEventsRequest =
FilterLogEventsRequest.builder()
                .logGroupName(logGroupName)

```



```

        .logStreamNames(logStreamName)
        .filterPattern(pattern)
        .build();

    FilterLogEventsResponse filterLogEventsResponse =
    cwlClient.filterLogEvents(filterLogEventsRequest);

    for (FilteredLogEvent event : filterLogEventsResponse.events()) {
        System.out.println(event.message());
    }

    System.out.println("-----"); //
    Separator for better readability
    }
}
}

```

Druckt Metadaten über den neuesten Protokollstream in einer angegebenen Protokollgruppe.

```

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CloudWatchLogQuery {
    public static void main(final String[] args) {
        final String usage = ""
            Usage:
            <logGroupName>

            Where:
            logGroupName - The name of the log group (for example, /aws/
            lambda/ChatAIHandler).
            "";

        if (args.length != 1) {

```

```
        System.out.print(usage);
        System.exit(1);
    }

    String logGroupName = "/aws/lambda/ChatAIHandler" ; //args[0];
    CloudWatchLogsClient logsClient = CloudWatchLogsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    describeMostRecentLogStream(logsClient, logGroupName);
}

/**
 * Describes and prints metadata about the most recent log stream in the
 * specified log group.
 *
 * @param logsClient the CloudWatchLogsClient used to interact with AWS
 * CloudWatch Logs
 * @param logGroupName the name of the log group
 */
public static void describeMostRecentLogStream(CloudWatchLogsClient
logsClient, String logGroupName) {
    DescribeLogStreamsRequest streamsRequest =
DescribeLogStreamsRequest.builder()
        .logGroupName(logGroupName)
        .orderBy(OrderBy.LAST_EVENT_TIME)
        .descending(true)
        .limit(1)
        .build();

    try {
        DescribeLogStreamsResponse streamsResponse =
logsClient.describeLogStreams(streamsRequest);
        List<LogStream> logStreams = streamsResponse.logStreams();

        if (logStreams.isEmpty()) {
            System.out.println("No log streams found for log group: " +
logGroupName);
            return;
        }

        LogStream stream = logStreams.get(0);
        System.out.println("Most Recent Log Stream:");
        System.out.println("  Name: " + stream.logStreamName());
    }
```

```
        System.out.println("  ARN: " + stream.arn());
        System.out.println("  Creation Time: " + stream.creationTime());
        System.out.println("  First Event Time: " +
stream.firstEventTimestamp());
        System.out.println("  Last Event Time: " +
stream.lastEventTimestamp());
        System.out.println("  Stored Bytes: " + stream.storedBytes());
        System.out.println("  Upload Sequence Token: " +
stream.uploadSequenceToken());

    } catch (CloudWatchLogsException e) {
        System.err.println("Failed to describe log stream: " +
e.awsErrorDetails().errorMessage());
    }
}
}
```

- Einzelheiten zur API finden Sie [DescribeLogStreams](#) unter AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

DescribeSubscriptionFilters Mit einem AWS SDK verwenden

Die folgenden Code-Beispiele zeigen, wie `DescribeSubscriptionFilters` verwendet wird.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DescribeSubscriptionFiltersRequest.h>
#include <aws/logs/model/DescribeSubscriptionFiltersResult.h>
#include <iostream>
#include <iomanip>
```

Listen Sie die Abonnementfilter auf.

```
Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::DescribeSubscriptionFiltersRequest request;
request.SetLogGroupName(log_group);
request.SetLimit(1);

bool done = false;
bool header = false;
while (!done) {
    auto outcome = cwl.DescribeSubscriptionFilters(
        request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to describe CloudWatch subscription filters
"
        << "for log group " << log_group << ": " <<
        outcome.GetError().GetMessage() << std::endl;
        break;
    }

    if (!header) {
        std::cout << std::left << std::setw(32) << "Name" <<
        std::setw(64) << "FilterPattern" << std::setw(64) <<
        "DestinationArn" << std::endl;
        header = true;
    }

    const auto &filters = outcome.GetResult().GetSubscriptionFilters();
    for (const auto &filter : filters) {
        std::cout << std::left << std::setw(32) <<
        filter.GetFilterName() << std::setw(64) <<
        filter.GetFilterPattern() << std::setw(64) <<
        filter.GetDestinationArn() << std::endl;
    }
}
```

```

        const auto &next_token = outcome.GetResult().GetNextToken();
        request.SetNextToken(next_token);
        done = next_token.empty();
    }

```

- Einzelheiten zur API finden Sie [DescribeSubscriptionFilters](#) in der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class DescribeSubscriptionFilters {
    public static void main(String[] args) {

```

```

    final String usage = ""

        Usage:
        <logGroup>

        Where:
        logGroup - A log group name (for example, myloggroup).
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String logGroup = args[0];
    CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    describeFilters(logs, logGroup);
    logs.close();
}

public static void describeFilters(CloudWatchLogsClient logs, String
logGroup) {
    try {
        boolean done = false;
        String newToken = null;

        while (!done) {
            DescribeSubscriptionFiltersResponse response;
            if (newToken == null) {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .logGroupName(logGroup)
                    .limit(1).build();

                response = logs.describeSubscriptionFilters(request);
            } else {
                DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                    .nextToken(newToken)
                    .logGroupName(logGroup)
                    .limit(1).build();

```

```
        response = logs.describeSubscriptionFilters(request);
    }

    for (SubscriptionFilter filter : response.subscriptionFilters())
    {
        System.out.printf("Retrieved filter with name %s, " +
            "pattern %s " + "and destination arn %s",
            filter.filterName(),
            filter.filterPattern(),
            filter.destinationArn());
    }

    if (response.nextToken() == null) {
        done = true;
    } else {
        newToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Done");
}
}
```

- Einzelheiten zur API finden Sie [DescribeSubscriptionFilters](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
import { DescribeSubscriptionFiltersCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  // This will return a list of all subscription filters in your account
  // matching the log group name.
  const command = new DescribeSubscriptionFiltersCommand({
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
    limit: 1,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Einzelheiten zur API finden Sie [DescribeSubscriptionFilters](#) in der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
```



```
    logGroupName: "GROUP_NAME",
    limit: 5,
  };

  cwl.describeSubscriptionFilters(params, function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data.subscriptionFilters);
    }
  });
});
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [DescribeSubscriptionFilters](#) in der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { cwlClient
->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name ${filter.filterName} pattern
${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

```
}  
}
```

- Einzelheiten zur API finden Sie [DescribeSubscriptionFilters](#) in der API-Referenz zum AWS SDK für Kotlin.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **GetLogEvents** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie GetLogEvents verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erstellen Sie Ihre erste Lambda-Funktion](#)

CLI

AWS CLI

Der folgende Befehl ruft Protokollereignisse aus einem Protokollstream mit dem Namen 20150601 in der Protokollgruppe my-logs ab:

```
aws logs get-log-events --log-group-name my-logs --log-stream-name 20150601
```

Ausgabe:

```
{  
  "nextForwardToken":  
    "f/31961209122447488583055879464742346735121166569214640130",  
  "events": [  
    {  
      "ingestionTime": 1433190494190,  
      "timestamp": 1433190184356,  
      "message": "Example Event 1"  
    },  
    {
```

```

        "ingestionTime": 1433190516679,
        "timestamp": 1433190184356,
        "message": "Example Event 1"
    },
    {
        "ingestionTime": 1433190494190,
        "timestamp": 1433190184358,
        "message": "Example Event 2"
    }
],
"nextBackwardToken":
"b/31961209122358285602261756944988674324553373268216709120"
}

```

- Einzelheiten zur API finden Sie [GetLogEvents](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsResponse;

import java.time.Instant;
import java.time.temporal.ChronoUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.

```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class GetLogEvents {

    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <logGroupName> <logStreamName>

            Where:
                logGroupName - The name of the log group (for example,
myloggroup).
                logStreamName - The name of the log stream (for example,
mystream).

            "";

        // if (args.length != 2) {
        //     System.out.print(usage);
        //     System.exit(1);
//     }

        String logGroupName = "WeathertopJavaContainerLogs" ; //args[0];
        String logStreamName = "weathertop-java-stream" ; //args[1];

        Region region = Region.US_EAST_1 ;
        CloudWatchLogsClient cloudWatchLogsClient =
CloudWatchLogsClient.builder()
            .region(region)
            .build();

        getCWLogEvents(cloudWatchLogsClient, logGroupName, logStreamName);
        cloudWatchLogsClient.close();
    }

    public static void getCWLogEvents(CloudWatchLogsClient cloudWatchLogsClient,
                                     String logGroupName,
                                     String logStreamPrefix) {
```

```
try {
    // First, find the exact log stream name
    DescribeLogStreamsRequest describeRequest =
DescribeLogStreamsRequest.builder()
        .logGroupName(logGroupName)
        .logStreamNamePrefix(logStreamPrefix)
        .limit(1) // get the first matching stream
        .build();

    DescribeLogStreamsResponse describeResponse =
cloudWatchLogsClient.describeLogStreams(describeRequest);

    if (describeResponse.logStreams().isEmpty()) {
        System.out.println("No matching log streams found for prefix: " +
logStreamPrefix);
        return;
    }

    String exactLogStreamName =
describeResponse.logStreams().get(0).logStreamName();
    System.out.println("Using exact log stream: " + exactLogStreamName);

    long startTime = Instant.now().minus(7,
ChronoUnit.DAYS).toEpochMilli();
    long endTime = Instant.now().toEpochMilli();

    GetLogEventsRequest getLogEventsRequest =
GetLogEventsRequest.builder()
        .logGroupName(logGroupName)
        .logStreamName(exactLogStreamName) // <-- exact name, not
prefix

        .startTime(startTime)
        .endTime(endTime)
        .startFromHead(true)
        .build();

    GetLogEventsResponse response =
cloudWatchLogsClient.getLogEvents(getLogEventsRequest);

    if (response.events().isEmpty()) {
        System.out.println("No log events found in the past 7 days.");
    } else {
        response.events().forEach(e -> System.out.println(e.message()));
    }
}
```

```
        } catch (CloudWatchException e) {  
            System.err.println(e.awsErrorDetails().errorMessage());  
            System.exit(1);  
        }  
    }  
}
```

- Einzelheiten zur API finden Sie [GetLogEvents](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

GetQueryResults Mit einem AWS SDK verwenden

Die folgenden Code-Beispiele zeigen, wie `GetQueryResults` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Ausführen einer umfangreichen Abfrage](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <summary>  
/// Gets the results of a CloudWatch Logs Insights query.  
/// </summary>  
/// <param name="queryId">The ID of the query.</param>  
/// <returns>The query results response.</returns>
```

```
public async Task<GetQueryResultsResponse?> GetQueryResultsAsync(string
queryId)
{
    try
    {
        var request = new GetQueryResultsRequest
        {
            QueryId = queryId
        };

        var response = await
        _amazonCloudWatchLogs.GetQueryResultsAsync(request);
        return response;
    }
    catch (ResourceNotFoundException ex)
    {
        _logger.LogError($"Query not found: {ex.Message}");
        return null;
    }
    catch (Exception ex)
    {
        _logger.LogError($"An error occurred while getting query results:
{ex.Message}");
        return null;
    }
}
```

- Einzelheiten zur API finden Sie [GetQueryResults](#) in der AWS SDK für .NET API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * Simple wrapper for the GetQueryResultsCommand.
```

```

    * @param {string} queryId
    */
    _getQueryResults(queryId) {
        return this.client.send(new GetQueryResultsCommand({ queryId }));
    }

```

- Einzelheiten zur API finden Sie [GetQueryResults](#) in der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

def _wait_for_query_results(self, client, query_id):
    """
    Waits for the query to complete and retrieves the results.

    :param query_id: The ID of the initiated query.
    :type query_id: str
    :return: A list containing the results of the query.
    :rtype: list
    """
    while True:
        time.sleep(1)
        results = client.get_query_results(queryId=query_id)
        if results["status"] in [
            "Complete",
            "Failed",
            "Cancelled",
            "Timeout",
            "Unknown",
        ]:
            return results.get("results", [])

```


- Einzelheiten zur API finden Sie [GetQueryResults](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
TRY.
    oo_result = lo_cwl->getqueryresults(
        iv_queryid = iv_query_id ).

    " Display query status and result count
    DATA(lv_status) = oo_result->get_status( ).
    DATA(lt_results) = oo_result->get_results( ).
    DATA(lv_result_count) = lines( lt_results ).

    MESSAGE |Query status: { lv_status }. Retrieved { lv_result_count } log
event(s).| TYPE 'I'.
    CATCH /aws1/cx_cwlinvalidparameterex.
        MESSAGE 'Invalid parameter.' TYPE 'E'.
    CATCH /aws1/cx_cwlresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'E'.
    CATCH /aws1/cx_cwlserviceunavailex.
        MESSAGE 'Service unavailable.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie [GetQueryResults](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

PutSubscriptionFilterMit einem AWS SDK verwenden

Die folgenden Code-Beispiele zeigen, wie PutSubscriptionFilter verwendet wird.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Erstellen Sie den Abonnementfilter.

```
Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::PutSubscriptionFilterRequest request;
request.SetFilterName(filter_name);
request.SetFilterPattern(filter_pattern);
request.SetLogGroupName(log_group);
request.SetDestinationArn(dest_arn);
auto outcome = cwl.PutSubscriptionFilter(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch logs subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
}
```

```

else
{
    std::cout << "Successfully created CloudWatch logs subscription " <<
        "filter " << filter_name << std::endl;
}

```

- Einzelheiten zur API finden Sie [PutSubscriptionFilter](#) in der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;

/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"
 * --source-account "111111111111"
 *
 * Make sure you replace the function name with your function name and replace
 * '111111111111' with your account details.
 * For more information, see "Subscription Filters with AWS Lambda" in the
 * Amazon CloudWatch Logs Guide.
 *
 *
 */

```

```

* Also, before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
*/

public class PutSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <filter> <pattern> <logGroup> <functionArn>\s

            Where:
                filter - A filter name (for example, myfilter).
                pattern - A filter pattern (for example, ERROR).
                logGroup - A log group name (testgroup).
                functionArn - An AWS Lambda function ARN (for example,
arn:aws:lambda:us-west-2:111111111111:function:lambda1) .
                "";

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String pattern = args[1];
        String logGroup = args[2];
        String functionArn = args[3];
        Region region = Region.US_WEST_2;
        CloudWatchLogsClient cwl = CloudWatchLogsClient.builder()
            .region(region)
            .build();

        putSubFilters(cwl, filter, pattern, logGroup, functionArn);
        cwl.close();
    }

    public static void putSubFilters(CloudWatchLogsClient cwl,

```

```
        String filter,
        String pattern,
        String logGroup,
        String functionArn) {

    try {
        PutSubscriptionFilterRequest request =
        PutSubscriptionFilterRequest.builder()
            .filterName(filter)
            .filterPattern(pattern)
            .logGroupName(logGroup)
            .destinationArn(functionArn)
            .build();

        cwl.putSubscriptionFilter(request);
        System.out.printf(
            "Successfully created CloudWatch logs subscription filter
%s",

            filter);

    } catch (CloudWatchLogsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Einzelheiten zur API finden Sie [PutSubscriptionFilter](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
import { PutSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutSubscriptionFilterCommand({
    // An ARN of a same-account Kinesis stream, Kinesis Firehose
    // delivery stream, or Lambda function.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
    SubscriptionFilters.html
    destinationArn: process.env.CLOUDWATCH_LOGS_DESTINATION_ARN,

    // A name for the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,

    // A filter pattern for subscribing to a filtered stream of log events.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
    FilterAndPatternSyntax.html
    filterPattern: process.env.CLOUDWATCH_LOGS_FILTER_PATTERN,

    // The name of the log group. Messages in this group matching the filter
    pattern
    // will be sent to the destination ARN.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Einzelheiten zur API finden Sie [PutSubscriptionFilter](#) in der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cw1 = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  destinationArn: "LAMBDA_FUNCTION_ARN",
  filterName: "FILTER_NAME",
  filterPattern: "ERROR",
  logGroupName: "LOG_GROUP",
};

cw1.putSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [PutSubscriptionFilter](#) in der AWS SDK für JavaScript API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

StartLiveTail Mit einem AWS SDK verwenden

Die folgenden Code-Beispiele zeigen, wie StartLiveTail verwendet wird.

.NET

SDK für .NET

Binden Sie die erforderlichen Dateien ein.

```
using Amazon;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
```

Starten Sie die Live-Tail-Sitzung.

```
var client = new AmazonCloudWatchLogsClient();
var request = new StartLiveTailRequest
{
    LogGroupIdentifiers = logGroupIdentifiers,
    LogStreamNames = logStreamNames,
    LogEventFilterPattern = filterPattern,
};

var response = await client.StartLiveTailAsync(request);

// Catch if request fails
if (response.HttpStatusCode != System.Net.HttpStatusCode.OK)
{
    Console.WriteLine("Failed to start live tail session");
    return;
}
```

Sie können die Ereignisse aus der Live Tail-Sitzung auf zwei Arten handhaben:

```
/* Method 1
 * 1). Asynchronously loop through the event stream
 * 2). Set a timer to dispose the stream and stop the Live Tail
session at the end.
*/
```



```

var eventStream = response.ResponseStream;
var task = Task.Run(() =>
{
    foreach (var item in eventStream)
    {
        if (item is LiveTailSessionUpdate liveTailSessionUpdate)
        {
            foreach (var sessionResult in
liveTailSessionUpdate.SessionResults)
            {
                Console.WriteLine("Message : {0}",
sessionResult.Message);
            }
        }
        if (item is LiveTailSessionStart)
        {
            Console.WriteLine("Live Tail session started");
        }
        // On-stream exceptions are processed here
        if (item is CloudWatchLogsEventStreamException)
        {
            Console.WriteLine($"ERROR: {item}");
        }
    }
});
// Close the stream to stop the session after a timeout
if (!task.Wait(TimeSpan.FromSeconds(10))){
    eventStream.Dispose();
    Console.WriteLine("End of line");
}

```

```

/* Method 2
* 1). Add event handlers to each event variable
* 2). Start processing the stream and wait for a timeout using
AutoResetEvent
*/
AutoResetEvent endEvent = new AutoResetEvent(false);
var eventStream = response.ResponseStream;
using (eventStream) // automatically disposes the stream to stop the
session after execution finishes
{
    eventStream.SessionStartReceived += (sender, e) =>

```

```

        {
            Console.WriteLine("LiveTail session started");
        };
        eventStream.SessionUpdateReceived += (sender, e) =>
        {
            foreach (LiveTailSessionLogEvent logEvent in
e.EventStreamEvent.SessionResults){
                Console.WriteLine("Message: {0}", logEvent.Message);
            }
        };
        // On-stream exceptions are captured here
        eventStream.ExceptionReceived += (sender, e) =>
        {
            Console.WriteLine($"ERROR:
{e.EventStreamException.Message}");
        };

        eventStream.StartProcessing();
        // Stream events for this amount of time.
        endEvent.WaitOne(TimeSpan.FromSeconds(10));
        Console.WriteLine("End of line");
    }
}

```

- Einzelheiten zur API finden Sie [StartLiveTail](#) in der AWS SDK für .NET API-Referenz.

Go

SDK für Go V2

Binden Sie die erforderlichen Dateien ein.

```

import (
    "context"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

```

Bearbeiten Sie die Ereignisse aus der Live-Tail-Sitzung.

```
func handleEventStreamAsync(stream *cloudwatchlogs.StartLiveTailEventStream) {
    eventsChan := stream.Events()
    for {
        event := <-eventsChan
        switch e := event.(type) {
        case *types.StartLiveTailResponseStreamMemberSessionStart:
            log.Println("Received SessionStart event")
        case *types.StartLiveTailResponseStreamMemberSessionUpdate:
            for _, logEvent := range e.Value.SessionResults {
                log.Println(*logEvent.Message)
            }
        default:
            // Handle on-stream exceptions
            if err := stream.Err(); err != nil {
                log.Fatalf("Error occurred during streaming: %v", err)
            } else if event == nil {
                log.Println("Stream is Closed")
                return
            } else {
                log.Fatalf("Unknown event type: %T", e)
            }
        }
    }
}
```

Starten Sie die Live-Tail-Sitzung.

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic("configuration error, " + err.Error())
}
client := cloudwatchlogs.NewFromConfig(cfg)

request := &cloudwatchlogs.StartLiveTailInput{
    LogGroupIdentifiers: logGroupIdentifiers,
    LogStreamNames:      logStreamNames,
    LogEventFilterPattern: logEventFilterPattern,
}

response, err := client.StartLiveTail(context.TODO(), request)
```

```
// Handle pre-stream Exceptions
if err != nil {
    log.Fatalf("Failed to start streaming: %v", err)
}

// Start a Goroutine to handle events over stream
stream := response.GetStream()
go handleEventStreamAsync(stream)
```

Beenden Sie die Live-Tail-Sitzung nach Ablauf einer gewissen Zeit.

```
// Close the stream (which ends the session) after a timeout
time.Sleep(10 * time.Second)
stream.Close()
log.Println("Event stream closed")
```

- Einzelheiten zur API finden Sie [StartLiveTail](#) unter AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Binden Sie die erforderlichen Dateien ein.

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import
    software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
```

```
import
    software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

Bearbeiten Sie die Ereignisse aus der Live-Tail-Sitzung.

```
private static StartLiveTailResponseHandler
getStartLiveTailResponseStreamHandler(
    AtomicReference<Subscription> subscriptionAtomicReference) {
    return StartLiveTailResponseHandler.builder()
        .onResponse(r -> System.out.println("Received initial response"))
        .onError(throwable -> {
            CloudWatchLogsException e = (CloudWatchLogsException)
throwable.getCause();
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        })
        .subscriber(() -> new FlowableSubscriber<>() {
            @Override
            public void onSubscribe(@NonNull Subscription s) {
                subscriptionAtomicReference.set(s);
                s.request(Long.MAX_VALUE);
            }

            @Override
            public void onNext(StartLiveTailResponseStream event) {
                if (event instanceof LiveTailSessionStart) {
                    LiveTailSessionStart sessionStart =
(LiveTailSessionStart) event;
                    System.out.println(sessionStart);
                } else if (event instanceof LiveTailSessionUpdate) {
                    LiveTailSessionUpdate sessionUpdate =
(LiveTailSessionUpdate) event;
                    List<LiveTailSessionLogEvent> logEvents =
sessionUpdate.sessionResults();
                    logEvents.forEach(e -> {
                        long timestamp = e.timestamp();
                        Date date = new Date(timestamp);
                        System.out.println "[" + date + " ] " + e.message());
                    });
                }
            }
        });
}
```

```

        });
    } else {
        throw CloudWatchLogsException.builder().message("Unknown
event type").build();
    }
}

@Override
public void onError(Throwable throwable) {
    System.out.println(throwable.getMessage());
    System.exit(1);
}

@Override
public void onComplete() {
    System.out.println("Completed Streaming Session");
}
})
.build();
}

```

Starten Sie die Live-Tail-Sitzung.

```

CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
    CloudWatchLogsAsyncClient.builder()
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

StartLiveTailRequest request =
    StartLiveTailRequest.builder()
        .logGroupIdentifiers(logGroupIdentifiers)
        .logStreamNames(logStreamNames)
        .logEventFilterPattern(logEventFilterPattern)
        .build();

/* Create a reference to store the subscription */
final AtomicReference<Subscription> subscriptionAtomicReference = new
AtomicReference<>(null);

cloudWatchLogsAsyncClient.startLiveTail(request,
getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));

```

Beenden Sie die Live-Tail-Sitzung nach Ablauf einer gewissen Zeit.

```
/* Set a timeout for the session and cancel the subscription. This will:  
 * 1). Close the stream  
 * 2). Stop the Live Tail session  
 */  
try {  
    Thread.sleep(10000);  
} catch (InterruptedException e) {  
    throw new RuntimeException(e);  
}  
if (subscriptionAtomicReference.get() != null) {  
    subscriptionAtomicReference.get().cancel();  
    System.out.println("Subscription to stream closed");  
}
```

- Einzelheiten zur API finden Sie [StartLiveTail](#) unter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Binden Sie die erforderlichen Dateien ein.

```
import { CloudWatchLogsClient, StartLiveTailCommand } from "@aws-sdk/client-cloudwatch-logs";
```

Bearbeiten Sie die Ereignisse aus der Live-Tail-Sitzung.

```
async function handleResponseAsync(response) {  
    try {  
        for await (const event of response.responseStream) {  
            if (event.sessionStart !== undefined) {  
                console.log(event.sessionStart);  
            } else if (event.sessionUpdate !== undefined) {  
                for (const logEvent of event.sessionUpdate.sessionResults) {  
                    const timestamp = logEvent.timestamp;  
                    const date = new Date(timestamp);  
                    console.log "[" + date + "]" + logEvent.message);  
                }  
            }  
        }  
    }  
}
```

```
    }
    } else {
        console.error("Unknown event type");
    }
}
} catch (err) {
    // On-stream exceptions are captured here
    console.error(err)
}
}
```

Starten Sie die Live-Tail-Sitzung.

```
const client = new CloudWatchLogsClient();

const command = new StartLiveTailCommand({
    logGroupIdentifiers: logGroupIdentifiers,
    logStreamNames: logStreamNames,
    logEventFilterPattern: filterPattern
});
try{
    const response = await client.send(command);
    handleResponseAsync(response);
} catch (err){
    // Pre-stream exceptions are captured here
    console.log(err);
}
```

Beenden Sie die Live-Tail-Sitzung nach Ablauf einer gewissen Zeit.

```
/* Set a timeout to close the client. This will stop the Live Tail session.
*/
setTimeout(function() {
    console.log("Client timeout");
    client.destroy();
}, 10000);
```

- Einzelheiten zur API finden Sie [StartLiveTail](#) in der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Binden Sie die erforderlichen Dateien ein.

```
import aws.sdk.kotlin.services.cloudwatchlogs.CloudWatchLogsClient
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailRequest
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailResponseStream
import kotlinx.coroutines.flow.takeWhile
```

Starten Sie die Live-Tail-Sitzung.

```
val client = CloudWatchLogsClient.fromEnvironment()

val request = StartLiveTailRequest {
    logGroupIdentifiers = logGroupIdentifiersVal
    logStreamNames = logStreamNamesVal
    logEventFilterPattern = logEventFilterPatternVal
}

val startTime = System.currentTimeMillis()

try {
    client.startLiveTail(request) { response ->
        val stream = response.responseStream
        if (stream != null) {
            /* Set a timeout to unsubscribe from the flow. This will:
             * 1). Close the stream
             * 2). Stop the Live Tail session
             */
            stream.takeWhile { System.currentTimeMillis() - startTime <
10000 }.collect { value ->
                if (value is StartLiveTailResponseStream.SessionStart) {
                    println(value.asSessionStart())
                } else if (value is
StartLiveTailResponseStream.SessionUpdate) {
                    for (e in value.asSessionUpdate().sessionResults!!) {
                        println(e)
                    }
                } else {
                    throw IllegalArgumentException("Unknown event type")
                }
            }
        }
    }
```

```

        }
    }
    } else {
        throw IllegalArgumentException("No response stream")
    }
}
} catch (e: Exception) {
    println("Exception occurred during StartLiveTail: $e")
    System.exit(1)
}

```

- Einzelheiten zur API finden Sie [StartLiveTail](#) in der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

Binden Sie die erforderlichen Dateien ein.

```

import boto3
import time
from datetime import datetime

```

Starten Sie die Live-Tail-Sitzung.

```

# Initialize the client
client = boto3.client('logs')

start_time = time.time()

try:
    response = client.start_live_tail(
        logGroupIdentifiers=log_group_identifiers,
        logStreamNames=log_streams,
        logEventFilterPattern=filter_pattern
    )
    event_stream = response['responseStream']
    # Handle the events streamed back in the response
    for event in event_stream:
        # Set a timeout to close the stream.

```

```

        # This will end the Live Tail session.
        if (time.time() - start_time >= 10):
            event_stream.close()
            break
        # Handle when session is started
        if 'sessionStart' in event:
            session_start_event = event['sessionStart']
            print(session_start_event)
        # Handle when log event is given in a session update
        elif 'sessionUpdate' in event:
            log_events = event['sessionUpdate']['sessionResults']
            for log_event in log_events:
                print('[{date}]'
{log}'.format(date=datetime.fromtimestamp(log_event['timestamp']/1000),log=log_event['me
            else:
                # On-stream exceptions are captured here
                raise RuntimeError(str(event))
    except Exception as e:
        print(e)

```

- Einzelheiten zur API finden Sie [StartLiveTail](#) in AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter.

[Verwenden von CloudWatch Logs mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

StartQueryMit einem AWS SDK verwenden

Die folgenden Code-Beispiele zeigen, wie StartQuery verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Ausführen einer umfangreichen Abfrage](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <summary>
/// Starts a CloudWatch Logs Insights query.
/// </summary>
/// <param name="logGroupName">The name of the log group to query.</param>
/// <param name="queryString">The CloudWatch Logs Insights query string.</
param>
/// <param name="startTime">The start time for the query (seconds since
epoch).</param>
/// <param name="endTime">The end time for the query (seconds since epoch).</
param>
/// <param name="limit">The maximum number of results to return.</param>
/// <returns>The query ID if successful, null otherwise.</returns>
public async Task<string?> StartQueryAsync(
    string logGroupName,
    string queryString,
    long startTime,
    long endTime,
    int limit = 10000)
{
    try
    {
        var request = new StartQueryRequest
        {
            LogGroupName = logGroupName,
            QueryString = queryString,
            StartTime = startTime,
            EndTime = endTime,
            Limit = limit
        };

        var response = await _amazonCloudWatchLogs.StartQueryAsync(request);
        return response.QueryId;
```

```

    }
    catch (InvalidParameterException ex)
    {
        _logger.LogError($"Invalid parameter for query: {ex.Message}");
        return null;
    }
    catch (ResourceNotFoundException ex)
    {
        _logger.LogError($"Log group not found: {ex.Message}");
        return null;
    }
    catch (Exception ex)
    {
        _logger.LogError($"An error occurred while starting query:
{ex.Message}");
        return null;
    }
}

```

- Einzelheiten zur API finden Sie [StartQuery](#) in der AWS SDK für .NET API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

/**
 * Wrapper for the StartQueryCommand. Uses a static query string
 * for consistency.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 * @returns {Promise<{ queryId: string }>}
 */
async _startQuery([startDate, endDate], maxLogs = 10000) {
    try {
        return await this.client.send(

```

```

        new StartQueryCommand({
            logGroupNames: this.logGroupNames,
            queryString: "fields @timestamp, @message | sort @timestamp asc",
            startTime: startDate.valueOf(),
            endTime: endDate.valueOf(),
            limit: maxLogs,
        }),
    );
} catch (err) {
    /** @type {string} */
    const message = err.message;
    if (message.startsWith("Query's end date and time")) {
        // This error indicates that the query's start or end date occur
        // before the log group was created.
        throw new DateOutOfBoundsError(message);
    }

    throw err;
}
}

```

- Einzelheiten zur API finden Sie [StartQuery](#) in der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

def perform_query(self, date_range):
    """
    Performs the actual CloudWatch log query.

    :param date_range: A tuple representing the start and end datetime for
    the query.
    :type date_range: tuple
    :return: A list containing the query results.
    """

```

```

        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_group,
                    startTime=start_time,
                    endTime=end_time,
                    queryString=self.query_string,
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
            except DateOutOfBoundsError:
                return []

def _initiate_query(self, client, date_range, max_logs):
    """
    Initiates the CloudWatch logs query.

    :param date_range: A tuple representing the start and end datetime for
    the query.
    :type date_range: tuple

```

```

        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")

```

- Einzelheiten zur API finden Sie [StartQuery](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

TRY.

```
" iv_log_group_name = '/aws/lambda/my-function'
```



```

    " iv_query_string = 'fields @timestamp, @message | sort @timestamp desc |
limit 20'
    " iv_start_time and iv_end_time must be in Unix epoch milliseconds (ms
since Jan 1, 1970 00:00:00 UTC)
    oo_result = lo_cwl->startquery(
        iv_loggroupname = iv_log_group_name
        iv_starttime     = iv_start_time
        iv_endtime       = iv_end_time
        iv_querystring   = iv_query_string
        iv_limit          = iv_limit ).

    " Display the query ID for tracking
    DATA(lv_query_id) = oo_result->get_queryid( ).
    MESSAGE |Query started successfully with ID: { lv_query_id }| TYPE 'I'.
CATCH /aws1/cx_cwlinvalidparameterex.
    MESSAGE 'Invalid parameter.' TYPE 'E'.
CATCH /aws1/cx_cwllimitexceededex.
    MESSAGE 'Limit exceeded.' TYPE 'E'.
CATCH /aws1/cx_cwlmalformedqueryex.
    MESSAGE 'Malformed query.' TYPE 'E'.
CATCH /aws1/cx_cwlresourcenotfoundex.
    MESSAGE 'Resource not found.' TYPE 'E'.
CATCH /aws1/cx_cwlserviceunavailex.
    MESSAGE 'Service unavailable.' TYPE 'E'.
ENDTRY.

```

- Einzelheiten zur API finden Sie [StartQuery](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Szenarien für CloudWatch Logs mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in CloudWatch Logs with implementieren AWS SDKs. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben ausführen können, indem Sie mehrere Funktionen in CloudWatch Logs aufrufen oder mit anderen kombinieren AWS-Services. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- [Amazon ECS Service Connect konfigurieren](#)
- [Erstellen Sie Ihre erste Lambda-Funktion](#)
- [Verwenden Sie CloudWatch Logs, um eine umfangreiche Abfrage auszuführen](#)
- [Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion](#)

Amazon ECS Service Connect konfigurieren

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie die VPC-Infrastruktur
- Einrichten der Protokollierung
- Erstellen Sie den ECS-Cluster
- Konfigurieren Sie IAM-Rollen
- Erstellen Sie den Dienst mit Service Connect
- Überprüfen der Bereitstellung
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im [Tutorials und Beispiele](#)-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
```

```
# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
```

```
# This script creates an ECS cluster with Service Connect and deploys an nginx
service
# Uses the default VPC to avoid VPC limits

set -e # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
```

```

    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
    "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
"$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
            done
        ;;
    esac
}

```

```

        ((attempt++))
    done
    log "ERROR: Cluster did not become active within expected time"
    return 1
    ;;
    "service")
        log "Waiting for service $resource_id to be stable..."
        aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
"$resource_id"
        ;;
    "nat-gateway")
        log "Waiting for NAT Gateway $resource_id to be available..."
        aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
        ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
"Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

```

```

log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

# Create security group for ECS tasks
SG_OUTPUT=$(aws ec2 create-security-group \
  --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
  --description "Security group for ECS Service Connect tutorial" \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
  check_for_errors "$SG_OUTPUT" "create-security-group"
  SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d' "' -f4)
  track_resource "SG:$SECURITY_GROUP_ID"
log "Created security group: $SECURITY_GROUP_ID"

# Add inbound rules to security group
aws ec2 authorize-security-group-ingress \
  --group-id "$SECURITY_GROUP_ID" \
  --protocol tcp \
  --port 80 \
  --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

aws ec2 authorize-security-group-ingress \
  --group-id "$SECURITY_GROUP_ID" \
  --protocol tcp \
  --port 443 \
  --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
  log "Creating CloudWatch log groups..."

  # Create log group for nginx container
  aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
  --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
"ResourceAlreadyExistsException" || {
    if [ ${PIPESTATUS[0]} -eq 0 ]; then
      log "Log group /ecs/service-connect-nginx created"
      track_resource "LOG_GROUP:/ecs/service-connect-nginx"
    else

```

```

        log "Log group /ecs/service-connect-nginx already exists"
    fi
}

# Create log group for service connect proxy
aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
--tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
"ResourceAlreadyExistsException" || {
    if [ ${PIPESTATUS[0]} -eq 0 ]; then
        log "Log group /ecs/service-connect-proxy created"
        track_resource "LOG_GROUP:/ecs/service-connect-proxy"
    else
        log "Log group /ecs/service-connect-proxy already exists"
    fi
}
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

```

```

    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version": "2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version": "2012-10-17",
    "Statement": [

```



```

        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

aws iam create-role \
    --role-name ecsTaskRole \
    --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
>/dev/null

    track_resource "IAM_ROLE:ecsTaskRole"
    aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
    log "Created ecsTaskRole"

    # Wait for role to be available
    sleep 10
fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",

```

```

        "portMappings": [
            {
                "containerPort": 80,
                "protocol": "tcp",
                "name": "nginx-port"
            }
        ],
        "essential": true,
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-nginx",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs"
            }
        }
    }
}
EOF

TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
"^"]*"' | cut -d'"' -f4)
track_resource "TASK_DEF:service-connect-nginx"
log "Created task definition: $TASK_DEF_ARN"

# Clean up temporary file
rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",

```

```
"desiredCount": 1,
"launchType": "FARGATE",
"networkConfiguration": {
  "awsvpcConfiguration": {
    "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
    "securityGroups": ["${SECURITY_GROUP_ID}"],
    "assignPublicIp": "ENABLED"
  }
},
"serviceConnectConfiguration": {
  "enabled": true,
  "namespace": "${NAMESPACE_NAME}",
  "services": [
    {
      "portName": "nginx-port",
      "discoveryName": "nginx",
      "clientAliases": [
        {
          "port": 80,
          "dnsName": "nginx"
        }
      ]
    }
  ],
  "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
      "awslogs-group": "/ecs/service-connect-proxy",
      "awslogs-region": "${REGION}",
      "awslogs-stream-prefix": "ecs-service-connect"
    }
  }
},
"tags": [
  {
    "key": "Environment",
    "value": "tutorial"
  },
  {
    "key": "project",
    "value": "doc-smith"
  },
  {
    "key": "tutorial",
```

```

        "value": "amazon-ecs-service-connect"
    }
]
}
EOF

SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
check_for_errors "$SERVICE_OUTPUT" "create-service"

track_resource "SERVICE:service-connect-nginx-service"
log "Created ECS service: service-connect-nginx-service"

wait_for_resource "service" "service-connect-nginx-service"

# Clean up temporary file
rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then

```

```

    log "Task ARN: $TASK_ARN"

    # Try to get task IP address
    TASK_IP=$(aws ecs describe-tasks \
        --cluster "$CLUSTER_NAME" \
        --tasks "$TASK_ARN" \
        --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`.value' \
        --output text 2>/dev/null || echo "")

    if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
        log "Task IP address: $TASK_IP"
    else
        log "Could not retrieve task IP address"
    fi
fi

# Check Service Connect namespace
NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
    --filters Name=TYPE,Values=HTTP \
    --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
    log "Service Connect namespace '$NAMESPACE_NAME' is active"
else
    log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
fi

# Display Service Connect configuration
log "Service Connect configuration:"
aws ecs describe-services \
    --cluster "$CLUSTER_NAME" \
    --services "service-connect-nginx-service" \
    --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "=====
    echo "CREATED RESOURCES"
    echo "=====
    for resource in "${CREATED_RESOURCES[@]}"; do

```

```

        echo "- $resource"
    done
    echo "======"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
"$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
"$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
"$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
"$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
>/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
"error" && log "Warning: Failed to delete role $resource_id"
                ;;
        esac
    done
}

```

```

        "IAM_ROLE")
        aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
        aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
"error" && log "Warning: Failed to delete role $resource_id"
        ;;
        "CLUSTER")
        aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
"error" && log "Warning: Failed to delete cluster $resource_id"
        ;;
        "SG")
        for attempt in 1 2 3 4 5; do
            if aws ec2 delete-security-group --group-id "$resource_id"
2>/dev/null; then
                break
            fi
            log "Security group $resource_id still has dependencies,
retrying in 30s ($attempt/5)..."
            sleep 30
        done
        ;;
        "LOG_GROUP")
        aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
        ;;
        "NAMESPACE")
        # First, delete any services in the namespace
        NAMESPACE_SERVICES=$(aws servicediscovery list-services \
            --filters Name=NAMESPACE_ID,Values="$resource_id" \
            --query 'Services[].Id' --output text 2>/dev/null || echo "")

        if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
"None" ]]; then
            for service_id in $NAMESPACE_SERVICES; do
                aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                sleep 2
            done
        fi

        # Then delete the namespace
        aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true

```

```
        ;;

    esac

    sleep 2 # Brief pause between deletions
done

# Clean up temporary files
rm -f /tmp/ecs-task-trust-policy.json
rm -f /tmp/task-definition.json
rm -f /tmp/service-definition.json

log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment
}
```



```

log "Tutorial completed successfully!"

# Display created resources
display_resources

# Ask user if they want to clean up
echo ""
echo "=====
echo "CLEANUP CONFIRMATION"
echo "=====
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    cleanup_resources
    log "All resources have been cleaned up"
else
    log "Resources left intact. You can clean them up later by running the
cleanup function."
    echo ""
    echo "To clean up resources later, you can use the AWS CLI commands or
the AWS Management Console."
    echo "Remember to delete resources in the correct order to avoid
dependency issues."
fi
}

# Make script executable and run
chmod +x "$0"
main "$@"

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
 - [AttachRolePolicy](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateCluster](#)
 - [CreateLogGroup](#)
 - [CreateRole](#)
 - [CreateSecurityGroup](#)

- [CreateService](#)
- [DeleteCluster](#)
- [DeleteLogGroup](#)
- [DeleteNamespace](#)
- [DeleteRole](#)
- [DeleteSecurityGroup](#)
- [DeleteService](#)
- [DeregisterTaskDefinition](#)
- [DescribeClusters](#)
- [DescribeServices](#)
- [DescribeSubnets](#)
- [DescribeTasks](#)
- [DescribeVpcs](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [GetRole](#)
- [ListNamespaces](#)
- [ListServices](#)
- [ListTaskDefinitions](#)
- [ListTasks](#)
- [RegisterTaskDefinition](#)
- [UpdateService](#)
- [Wait](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erstellen Sie Ihre erste Lambda-Funktion

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

Erstellen Sie Ihre erste Lambda-Funktion

- Erstellen Sie eine IAM-Rolle für Lambda
- Funktionscode erstellen
- Erstellen einer Lambda-Funktion
- Testen Sie Ihre Lambda-Funktion
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im [Tutorials und Beispiele](#)-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

#####
# Setup
#####

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"
```

```

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES

#####
# Helper functions
#####

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"

        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))

```

```

        if [ "$DELETE_WAIT" -ge 60 ]; then
            echo "  WARNING: Timed out waiting for function
deletion."
            break
        fi
    done
    ;;
    iam-role-policy)
        local ROLE_PART="${NAME%|*}"
        local POLICY_PART="${NAME#*|}"
        echo "Detaching policy from role: ${ROLE_PART}"
        aws iam detach-role-policy \
            --role-name "$ROLE_PART" \
            --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could
not detach policy from role ${ROLE_PART}."
        ;;
    iam-role)
        echo "Deleting IAM role: ${NAME}"
        aws iam delete-role \
            --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
IAM role ${NAME}."
        ;;
    esac
done

if [ -d "$TEMP_DIR" ]; then
    rm -rf "$TEMP_DIR"
fi

echo ""
echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "====="
    echo "ERROR: Script failed at $1"
    echo "====="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

```

```

}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

#####
# Region pre-check
#####

```

```

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
"$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi

#####
# Runtime selection
#####

echo ""
echo "=====
echo "AWS Lambda - Create Your First Function"
echo "=====
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in
event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)

```

```

        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
PYTHON_EOF
    echo "Selected runtime: Python 3.13"
    ;;
2)
    RUNTIME="nodejs22.x"
    HANDLER="index.handler"
    CODE_FILE="index.mjs"
    cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
    if (!event || typeof event.length !== 'number' || typeof event.width !==
'number') {
        throw new Error('Event must contain numeric length and width');
    }
    if (event.length < 0 || event.width < 0) {
        throw new Error('Length and width must be non-negative');
    }
    const area = event.length * event.width;
    console.log(`The area is ${area}`);
    console.log('CloudWatch log group: ', context.logGroupName);
    return JSON.stringify({area});
};
NODEJS_EOF
    echo "Selected runtime: Node.js 22.x"
    ;;
*)
    echo "ERROR: Invalid choice. Please enter 1 or 2."
    exit 1
    ;;
esac

#####
# Step 1: Create IAM execution role
#####

echo ""
echo "=====
```



```
echo "Step 1: Create IAM execution role"
echo "====="
echo ""

TRUST_POLICY='{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "${ROLE_NAME}" \
    --assume-role-policy-document "${TRUST_POLICY}" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "${ROLE_OUTPUT}" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi

echo "${ROLE_OUTPUT}"
ROLE_ARN="${ROLE_OUTPUT}"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "${ROLE_NAME}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "${ROLE_NAME}" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole" 2>&1
```

```

CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/
service-role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

#####
# Step 2: Create Lambda function
#####

echo ""
echo "=====
echo "Step 2: Create Lambda function"
echo "=====
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \

```

```

--tags project=doc-smith,tutorial=lambda-gettingstarted \
--query '[FunctionName, FunctionArn, Runtime, State]' \
--output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi

echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --
query State --output text" \
    "Active"

#####
# Step 3: Invoke the function
#####

echo ""
echo "=====
echo "Step 3: Invoke the function"
echo "=====
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

```

```

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

#####
# Step 4: View CloudWatch logs
#####

echo ""
echo "=====
echo "Step 4: View CloudWatch Logs"
echo "=====
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "${LOG_GROUP_NAME}" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

```

```

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

#####
# Summary and cleanup
#####

echo ""
echo "=====
echo "SUMMARY"
echo "=====
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "=====
echo "CLEANUP"
echo "=====
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

```

```
echo ""  
echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
 - [AttachRolePolicy](#)
 - [CreateFunction](#)
 - [CreateRole](#)
 - [DeleteFunction](#)
 - [DeleteLogGroup](#)
 - [DeleteRole](#)
 - [DescribeLogStreams](#)
 - [DetachRolePolicy](#)
 - [GetFunction](#)
 - [GetFunctionConfiguration](#)
 - [GetLogEvents](#)
 - [Invoke](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie CloudWatch Logs, um eine umfangreiche Abfrage auszuführen

Die folgenden Codebeispiele zeigen, wie CloudWatch Logs verwendet werden kann, um mehr als 10.000 Datensätze abzufragen.

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Dies ist der Hauptworkflow, der das Szenario mit großen Abfragen demonstriert.

```
using System.Diagnostics;
using System.Text.RegularExpressions;
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
using CloudWatchLogsActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace CloudWatchLogsScenario;

public class LargeQueryWorkflow
{
    /*
        Before running this .NET code example, set up your development environment,
        including your credentials.

        This .NET code example performs the following tasks for the CloudWatch Logs
        Large Query workflow:

        1. Prepare the Application:
            - Prompt the user to deploy CloudFormation stack and generate sample logs.
            - Deploy the CloudFormation template for resource creation.
            - Generate 50,000 sample log entries using CloudWatch Logs API.
            - Wait 5 minutes for logs to be fully ingested.

        2. Execute Large Query:
            - Perform recursive queries to retrieve all logs using binary search.
            - Display progress for each query executed.
            - Show total execution time and logs found.
```

```

3. Clean up:
    - Prompt the user to delete the CloudFormation stack and all resources.
    - Destroy the CloudFormation stack and wait until removed.
*/

public static ILogger<LargeQueryWorkflow> _logger = null!;
public static CloudWatchLogsWrapper _wrapper = null!;
public static IAmazonCloudFormation _amazonCloudFormation = null!;

private static string _logGroupName = "/workflows/cloudwatch-logs/large-
query";
private static string _logStreamName = "stream1";
private static long _queryStartDate;
private static long _queryEndDate;

public static bool _interactive = true;
public static string _stackName = "CloudWatchLargeQueryStack";
private static string _stackResourcePath = "../..../..../..../scenarios/
features/cloudwatch_logs_large_query/resources/stack.yaml";

public static async Task Main(string[] args)
{
    using var host = Host.CreateDefaultBuilder(args)
        .ConfigureLogging(logging =>
            logging.AddFilter("System", LogLevel.Debug)
                .AddFilter("Microsoft", LogLevel.Information))
        .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonCloudWatchLogs>()
                .AddAWSService<IAmazonCloudFormation>()
                .AddTransient<CloudWatchLogsWrapper>()
            )
        .Build();

    if (_interactive)
    {
        _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<LargeQueryWorkflow>();

        _wrapper = host.Services.GetRequiredService<CloudWatchLogsWrapper>();
        _amazonCloudFormation =
host.Services.GetRequiredService<IAmazonCloudFormation>();
    }
}

```



```

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query
Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to
retrieve more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

    try
    {
        Console.WriteLine(new string('-', 80));
        var prepareSuccess = await PrepareApplication();
        Console.WriteLine(new string('-', 80));

        if (prepareSuccess)
        {
            Console.WriteLine(new string('-', 80));
            await ExecuteLargeQuery();
            Console.WriteLine(new string('-', 80));
        }

        Console.WriteLine(new string('-', 80));
        await Cleanup();
        Console.WriteLine(new string('-', 80));
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "There was a problem with the scenario,
initiating cleanup...");
        _interactive = false;
        await Cleanup();
    }

    Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
}

/// <summary>
/// Runs the scenario workflow. Used for testing.
/// </summary>
public static async Task RunScenario()
{

```

```

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query
Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to
retrieve more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await ExecuteLargeQuery();
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario,
initiating cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
    }

    /// <summary>
    /// Prepares the application by creating the necessary resources.
    /// </summary>
    /// <returns>True if the application was prepared successfully.</returns>
    public static async Task<bool> PrepareApplication()

```

```
{
    Console.WriteLine("Preparing the application...");
    Console.WriteLine();

    try
    {
        var deployStack = !_interactive || GetYesNoResponse(
            "Would you like to deploy the CloudFormation stack and generate
sample logs? (y/n) ");

        if (deployStack)
        {
            if (_interactive)
            {
                Console.Write(
                    $"Enter a path for the CloudFormation stack
resource .yaml file (or press Enter for default '{_stackResourcePath}'): ");
                string? inputPath = Console.ReadLine();
                if (!string.IsNullOrEmpty(inputPath))
                {
                    _stackResourcePath = inputPath;
                }
            }

            _stackName = PromptUserForStackName();

            var deploySuccess = await DeployCloudFormationStack(_stackName);

            if (deploySuccess)
            {
                Console.WriteLine();
                Console.WriteLine("Generating 50,000 sample log entries...");
                var generateSuccess = await GenerateSampleLogs();

                if (generateSuccess)
                {
                    Console.WriteLine();
                    Console.WriteLine("Sample logs created. Waiting 5 minutes
for logs to be fully ingested...");
                    await WaitWithCountdown(300);

                    Console.WriteLine("Application preparation complete.");
                    return true;
                }
            }
        }
    }
}
```

```

        }
    }
    else
    {
        _logGroupName = PromptUserForInput("Enter the log group name ",
        _logGroupName);
        _logStreamName = PromptUserForInput("Enter the log stream name ",
        _logStreamName);

        var startDateMs = PromptUserForLong("Enter the query start date
(millisecons since epoch): ");
        var endDateMs = PromptUserForLong("Enter the query end date
(millisecons since epoch): ");

        _queryStartDate = startDateMs / 1000;
        _queryEndDate = endDateMs / 1000;

        Console.WriteLine("Application preparation complete.");
        return true;
    }
}
catch (Exception ex)
{
    _logger.LogError(ex, "An error occurred while preparing the
application.");
}

Console.WriteLine("Application preparation failed.");
return false;
}

/// <summary>
/// Deploys the CloudFormation stack with the necessary resources.
/// </summary>
/// <param name="stackName">The name of the CloudFormation stack.</param>
/// <returns>True if the stack was deployed successfully.</returns>
private static async Task<bool> DeployCloudFormationStack(string stackName)
{
    Console.WriteLine($"\\nDeploying CloudFormation stack: {stackName}");

    try
    {
        var request = new CreateStackRequest
        {

```

```
        StackName = stackName,
        TemplateBody = await File.ReadAllTextAsync(_stackResourcePath)
    };

    var response = await _amazonCloudFormation.CreateStackAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

        bool stackCreated = await
WaitForStackCompletion(response.StackId);

        if (stackCreated)
        {
            Console.WriteLine("CloudFormation stack created
successfully.");
            return true;
        }
        else
        {
            _logger.LogError($"CloudFormation stack creation failed:
{stackName}");
            return false;
        }
    }
    else
    {
        _logger.LogError($"Failed to create CloudFormation stack:
{stackName}");
        return false;
    }
}
catch (AlreadyExistsException)
{
    _logger.LogWarning($"CloudFormation stack '{stackName}' already
exists. Please provide a unique name.");
    var newStackName = PromptUserForStackName();
    return await DeployCloudFormationStack(newStackName);
}
catch (Exception ex)
{

```

```
        _logger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
        return false;
    }
}

/// <summary>
/// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
/// </summary>
/// <param name="stackId">The ID of the CloudFormation stack.</param>
/// <returns>True if the stack was created successfully.</returns>
private static async Task<bool> WaitForStackCompletion(string stackId)
{
    int retryCount = 0;
    const int maxRetries = 30;
    const int retryDelay = 10000;

    while (retryCount < maxRetries)
    {
        var describeStacksRequest = new DescribeStacksRequest
        {
            StackName = stackId
        };

        var describeStacksResponse = await
        _amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

        if (describeStacksResponse.Stacks.Count > 0)
        {
            if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
            {
                return true;
            }
            if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
            {
                return false;
            }
        }
    }
}
```

```

        Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
        await Task.Delay(retryDelay);
        retryCount++;
    }

    _logger.LogError("Timed out waiting for CloudFormation stack creation to
complete.");
    return false;
}

/// <summary>
/// Generates sample logs directly using CloudWatch Logs API.
/// Creates 50,000 log entries spanning 5 minutes.
/// </summary>
/// <returns>True if logs were generated successfully.</returns>
private static async Task<bool> GenerateSampleLogs()
{
    const int totalEntries = 50000;
    const int entriesPerBatch = 10000;
    const int fiveMinutesMs = 5 * 60 * 1000;

    try
    {
        // Calculate timestamps
        var startTimeMs = DateTimeOffset.UtcNow.ToUnixTimeMilliseconds();
        var timestampIncrement = fiveMinutesMs / totalEntries;

        Console.WriteLine($"Generating {totalEntries} log entries...");

        var entryCount = 0;
        var currentTimestamp = startTimeMs;
        var numBatches = totalEntries / entriesPerBatch;

        // Generate and upload logs in batches
        for (int batchNum = 0; batchNum < numBatches; batchNum++)
        {
            var logEvents = new List<InputLogEvent>();

            for (int i = 0; i < entriesPerBatch; i++)
            {
                logEvents.Add(new InputLogEvent
                {

```

```

        Timestamp =
DateTimeOffset.FromUnixTimeMilliseconds(currentTimestamp).UtcDateTime,
        Message = $"Entry {entryCount}"
    });

    entryCount++;
    currentTimestamp += timestampIncrement;
}

// Upload batch
var success = await _wrapper.PutLogEventsAsync(_logGroupName,
_logStreamName, logEvents);
if (!success)
{
    _logger.LogError($"Failed to upload batch {batchNum + 1}/{
{numBatches}");
    return false;
}

    Console.WriteLine($"Uploaded batch {batchNum + 1}/{numBatches}");
}

// Set query date range (convert milliseconds to seconds for query
API)
_queryStartDate = startTimeMs / 1000;
_queryEndDate = (currentTimestamp - timestampIncrement) / 1000;

    Console.WriteLine($"Query start date:
{DateTimeOffset.FromUnixTimeSeconds(_queryStartDate):yyyy-MM-
ddTHH:mm:ss.fffZ}");
    Console.WriteLine($"Query end date:
{DateTimeOffset.FromUnixTimeSeconds(_queryEndDate):yyyy-MM-ddTHH:mm:ss.fffZ}");
    Console.WriteLine($"Successfully uploaded {totalEntries} log
entries");

    return true;
}
catch (Exception ex)
{
    _logger.LogError(ex, "An error occurred while generating sample
logs.");
    return false;
}
}

```



```
/// <summary>
/// Executes the large query workflow.
/// </summary>
public static async Task ExecuteLargeQuery()
{
    Console.WriteLine("Starting recursive query to retrieve all logs...");
    Console.WriteLine();

    var queryLimit = PromptUserForInteger("Enter the query limit (max 10000)
", 10000);
    if (queryLimit > 10000) queryLimit = 10000;

    var queryString = "fields @timestamp, @message | sort @timestamp asc";

    var stopwatch = Stopwatch.StartNew();
    var allResults = await PerformLargeQuery(_logGroupName, queryString,
_queryStartDate, _queryEndDate, queryLimit);
    stopwatch.Stop();

    Console.WriteLine();
    Console.WriteLine($"Queries finished in
{stopwatch.Elapsed.TotalSeconds:F3} seconds.");
    Console.WriteLine($"Total logs found: {allResults.Count}");

    // Check for duplicates
    Console.WriteLine();
    Console.WriteLine("Checking for duplicate logs...");
    var duplicates = FindDuplicateLogs(allResults);
    if (duplicates.Count > 0)
    {
        Console.WriteLine($"WARNING: Found {duplicates.Count} duplicate log
entries!");
        Console.WriteLine("Duplicate entries (showing first 10):");
        foreach (var dup in duplicates.Take(10))
        {
            Console.WriteLine($" [{dup.Timestamp}] {dup.Message} (appears
{dup.Count} times)");
        }

        var uniqueCount = allResults.Count - duplicates.Sum(d => d.Count -
1);
        Console.WriteLine($"Unique logs: {uniqueCount}");
    }
}
```

```

        else
        {
            Console.WriteLine("No duplicates found. All logs are unique.");
        }
        Console.WriteLine();

        var viewSample = !_interactive || GetYesNoResponse("Would you like to see
a sample of the logs? (y/n) ");
        if (viewSample)
        {
            Console.WriteLine();
            Console.WriteLine($"Sample logs (first 10 of {allResults.Count}):");
            for (int i = 0; i < Math.Min(10, allResults.Count); i++)
            {
                var timestamp = allResults[i].Find(f => f.Field ==
"@timestamp")?.Value ?? "N/A";
                var message = allResults[i].Find(f => f.Field ==
"@message")?.Value ?? "N/A";
                Console.WriteLine($"[{timestamp}] {message}");
            }
        }
    }

    /// <summary>
    /// Performs a large query using recursive binary search.
    /// </summary>
    private static async Task<List<List<ResultField>>> PerformLargeQuery(
        string logGroupName,
        string queryString,
        long startTime,
        long endTime,
        int limit)
    {
        var queryId = await _wrapper.StartQueryAsync(logGroupName, queryString,
startTime, endTime, limit);
        if (queryId == null)
        {
            return new List<List<ResultField>>>();
        }

        var results = await PollQueryResults(queryId);
        if (results == null || results.Count == 0)
        {
            return new List<List<ResultField>>>();
        }
    }

```

```
}

    var startDate =
        DateTimeOffset.FromUnixTimeSeconds(startTime).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
    var endDate = DateTimeOffset.FromUnixTimeSeconds(endTime).ToString("yyyy-
MM-ddTHH:mm:ss.fffZ");
    Console.WriteLine($"Query date range: {startDate} ({startTime}s) to
{endDate} ({endTime}s). Found {results.Count} logs.");

    if (results.Count < limit)
    {
        Console.WriteLine($" -> Returning {results.Count} logs (less than
limit of {limit})");
        return results;
    }

    Console.WriteLine($" -> Hit limit of {limit}. Need to split and
recurse.");

    // Get the timestamp of the last log (sorted to find the actual last one)
    var lastLogTimestamp = GetLastLogTimestamp(results);
    if (lastLogTimestamp == null)
    {
        Console.WriteLine($" -> No timestamp found in results. Returning
{results.Count} logs.");
        return results;
    }

    Console.WriteLine($" -> Last log timestamp: {lastLogTimestamp}");

    // Parse the timestamp and add 1 millisecond to avoid querying the same
log again
    var lastLogDate = DateTimeOffset.Parse(lastLogTimestamp + " +0000");
    Console.WriteLine($" -> Last log as DateTimeOffset: {lastLogDate:yyyy-
MM-ddTHH:mm:ss.fffZ} ({lastLogDate.ToUnixTimeSeconds()}s)");

    var offsetLastLogDate = lastLogDate.AddMilliseconds(1);
    Console.WriteLine($" -> Offset timestamp (last
+ 1ms): {offsetLastLogDate:yyyy-MM-ddTHH:mm:ss.fffZ}
({offsetLastLogDate.ToUnixTimeSeconds()}s)");

    // Convert to seconds, but round UP to the next second to avoid
overlapping with logs in the same second
```

```
// This ensures we don't re-query logs that share the same second as the
last log
var offsetLastLogTime = offsetLastLogDate.ToUnixTimeSeconds();
if (offsetLastLogDate.Millisecond > 0)
{
    offsetLastLogTime++; // Move to the next full second
    Console.WriteLine($" -> Adjusted to next full second:
{offsetLastLogTime}s
({DateTimeOffset.FromUnixTimeSeconds(offsetLastLogTime):yyyy-MM-
ddTHH:mm:ss.fffZ})");
}

Console.WriteLine($" -> Comparing:
offsetLastLogTime={offsetLastLogTime}s vs endTime={endTime}s");
Console.WriteLine($" -> End time as date:
{DateTimeOffset.FromUnixTimeSeconds(endTime):yyyy-MM-ddTHH:mm:ss.fffZ}");

// Check if there's any time range left to query
if (offsetLastLogTime >= endTime)
{
    Console.WriteLine($" -> No time range left to query. Offset time
({offsetLastLogTime}s) >= end time ({endTime}s)");
    return results;
}

// Split the remaining date range in half
var (range1Start, range1End, range2Start, range2End) =
SplitDateRange(offsetLastLogTime, endTime);

var range1StartDate =
DateTimeOffset.FromUnixTimeSeconds(range1Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
var range1EndDate =
DateTimeOffset.FromUnixTimeSeconds(range1End).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
var range2StartDate =
DateTimeOffset.FromUnixTimeSeconds(range2Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
var range2EndDate =
DateTimeOffset.FromUnixTimeSeconds(range2End).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");

Console.WriteLine($" -> Splitting remaining range:");
```

```
        Console.WriteLine($"    Range 1: {range1StartDate} ({range1Start}s) to  
        {range1EndDate} ({range1End}s)");  
        Console.WriteLine($"    Range 2: {range2StartDate} ({range2Start}s) to  
        {range2EndDate} ({range2End}s)");  
  
        // Query both halves recursively  
        Console.WriteLine($"    -> Querying range 1...");  
        var results1 = await PerformLargeQuery(logGroupName, queryString,  
range1Start, range1End, limit);  
        Console.WriteLine($"    -> Range 1 returned {results1.Count} logs");  
  
        Console.WriteLine($"    -> Querying range 2...");  
        var results2 = await PerformLargeQuery(logGroupName, queryString,  
range2Start, range2End, limit);  
        Console.WriteLine($"    -> Range 2 returned {results2.Count} logs");  
  
        // Combine all results  
        var allResults = new List<List<ResultField>>(results);  
        allResults.AddRange(results1);  
        allResults.AddRange(results2);  
  
        Console.WriteLine($"    -> Combined total: {allResults.Count} logs  
        ({results.Count} + {results1.Count} + {results2.Count})");  
  
        return allResults;  
    }  
  
    /// <summary>  
    /// Gets the timestamp string of the most recent log from a list of logs.  
    /// Sorts timestamps to find the actual last one.  
    /// </summary>  
    private static string? GetLastLogTimestamp(List<List<ResultField>> logs)  
    {  
        var timestamps = logs  
            .Select(log => log.Find(f => f.Field == "@timestamp")?.Value)  
            .Where(t => !string.IsNullOrEmpty(t))  
            .OrderBy(t => t)  
            .ToList();  
  
        if (timestamps.Count == 0)  
        {  
            return null;  
        }  
    }
```

```
        return timestamps[timestamps.Count - 1];
    }

    /// <summary>
    /// Splits a date range in half.
    /// Range 2 starts at midpoint + 1 second to avoid overlap.
    /// </summary>
    private static (long range1Start, long range1End, long range2Start, long
range2End) SplitDateRange(long startTime, long endTime)
    {
        var midpoint = startTime + (endTime - startTime) / 2;
        // Range 2 starts at midpoint + 1 to avoid querying the same second twice
        return (startTime, midpoint, midpoint + 1, endTime);
    }

    /// <summary>
    /// Polls for query results until complete.
    /// </summary>
    private static async Task<List<List<ResultField>>>> PollQueryResults(string
queryId)
    {
        int retryCount = 0;
        const int maxRetries = 60;
        const int retryDelay = 1000;

        while (retryCount < maxRetries)
        {
            var response = await _wrapper.GetQueryResultsAsync(queryId);
            if (response == null)
            {
                return null;
            }

            if (response.Status == QueryStatus.Complete)
            {
                return response.Results;
            }

            if (response.Status == QueryStatus.Failed ||
                response.Status == QueryStatus.Cancelled ||
                response.Status == QueryStatus.Timeout ||
                response.Status == QueryStatus.Unknown)
            {
                _logger.LogError($"Query failed with status: {response.Status}");
            }
        }
    }
}
```

```
        return null;
    }

    await Task.Delay(retryDelay);
    retryCount++;
}

_logger.LogError("Timed out waiting for query results.");
return null;
}

/// <summary>
/// Cleans up the resources created during the scenario.
/// </summary>
public static async Task<bool> Cleanup()
{
    var cleanup = !_interactive || GetYesNoResponse(
        "Do you want to delete the CloudFormation stack and all resources?
(y/n) ");

    if (cleanup)
    {
        try
        {
            var stackDeleteSuccess = await
DeleteCloudFormationStack(_stackName, false);
            return stackDeleteSuccess;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while cleaning up the
resources.");
            return false;
        }
    }

    Console.WriteLine($"Resources will remain. Stack name: {_stackName}, Log
group: {_logGroupName}");
    _logger.LogInformation("CloudWatch Logs Large Query scenario is
complete.");
    return true;
}

/// <summary>
```

```
/// Deletes the CloudFormation stack and waits for confirmation.
/// </summary>
private static async Task<bool> DeleteCloudFormationStack(string stackName,
bool forceDelete)
{
    var request = new DeleteStackRequest
    {
        StackName = stackName,
    };

    if (forceDelete)
    {
        request.DeletionMode = DeletionMode.FORCE_DELETE_STACK;
    }

    await _amazonCloudFormation.DeleteStackAsync(request);
    Console.WriteLine($"CloudFormation stack '{stackName}' is being deleted.
This may take a few minutes.");

    bool stackDeleted = await WaitForStackDeletion(stackName, forceDelete);

    if (stackDeleted)
    {
        Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
        return true;
    }
    else
    {
        _logger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
        return false;
    }
}

/// <summary>
/// Waits for the stack to be deleted.
/// </summary>
private static async Task<bool> WaitForStackDeletion(string stackName, bool
forceDelete)
{
    int retryCount = 0;
    const int maxRetries = 30;
    const int retryDelay = 10000;
```



```
while (retryCount < maxRetries)
{
    var describeStacksRequest = new DescribeStacksRequest
    {
        StackName = stackName
    };

    try
    {
        var describeStacksResponse = await
        _amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

        if (describeStacksResponse.Stacks.Count == 0 ||
            describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
        {
            return true;
        }

        if (!forceDelete && describeStacksResponse.Stacks[0].StackStatus
== StackStatus.DELETE_FAILED)
        {
            return await DeleteCloudFormationStack(stackName, true);
        }
    }
    catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
    {
        return true;
    }

    Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to
be deleted...");
    await Task.Delay(retryDelay);
    retryCount++;
}

_logger.LogError($"Timed out waiting for CloudFormation stack
'{stackName}' to be deleted.");
return false;
}

/// <summary>
```

```

    /// Waits with a countdown display.
    /// </summary>
    private static async Task WaitWithCountdown(int seconds)
    {
        for (int i = seconds; i > 0; i--)
        {
            Console.Write($"\\rWaiting: {i} seconds remaining... ");
            await Task.Delay(1000);
        }
        Console.WriteLine("\\rWait complete.                ");
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompts the user for a stack name.
    /// </summary>
    private static string PromptUserForStackName()
    {
        if (_interactive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press
Enter for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrEmpty(input))
            {
                var regex = "[a-zA-Z][-a-zA-Z0-9]*";
                if (!Regex.IsMatch(input, regex))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
{_stackName}");
                    return _stackName;
                }
                return input;
            }
        }
    }

```

```
        }
    }
    return _stackName;
}

/// <summary>
/// Prompts the user for input with a default value.
/// </summary>
private static string PromptUserForInput(string prompt, string defaultValue)
{
    if (_interactive)
    {
        Console.Write($"{prompt}(press Enter for default '{defaultValue}'):");

        string? input = Console.ReadLine();
        return string.IsNullOrEmpty(input) ? defaultValue : input;
    }
    return defaultValue;
}

/// <summary>
/// Prompts the user for an integer value.
/// </summary>
private static int PromptUserForInteger(string prompt, int defaultValue)
{
    if (_interactive)
    {
        Console.Write($"{prompt}(press Enter for default '{defaultValue}'):");

        string? input = Console.ReadLine();
        if (string.IsNullOrEmpty(input) || !int.TryParse(input, out var result))
        {
            return defaultValue;
        }
        return result;
    }
    return defaultValue;
}

/// <summary>
/// Prompts the user for a long value.
/// </summary>
private static long PromptUserForLong(string prompt)
```

```
{
    if (_interactive)
    {
        Console.Write(prompt);
        string? input = Console.ReadLine();
        if (long.TryParse(input, out var result))
        {
            return result;
        }
    }
    return 0;
}

/// <summary>
/// Finds duplicate log entries based on timestamp and message.
/// </summary>
private static List<(string Timestamp, string Message, int Count)>
FindDuplicateLogs(List<List<ResultField>> logs)
{
    var logSignatures = new Dictionary<string, int>();

    foreach (var log in logs)
    {
        var timestamp = log.Find(f => f.Field == "@timestamp")?.Value ?? "";
        var message = log.Find(f => f.Field == "@message")?.Value ?? "";
        var signature = $"{timestamp}|{message}";

        if (logSignatures.ContainsKey(signature))
        {
            logSignatures[signature]++;
        }
        else
        {
            logSignatures[signature] = 1;
        }
    }

    return logSignatures
        .Where(kvp => kvp.Value > 1)
        .Select(kvp =>
        {
            var parts = kvp.Key.Split('|');
            return (Timestamp: parts[0], Message: parts[1], Count:
kvp.Value);
        });
}
```

```
        })
        .OrderByDescending(x => x.Count)
        .ToList();
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
 - [GetQueryResults](#)
 - [StartQuery](#)

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel](#) einrichten und ausführen.

Dies ist der Einstiegspunkt.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
import { CloudWatchQuery } from "./cloud-watch-query.js";

console.log("Starting a recursive query...");

if (!process.env.QUERY_START_DATE || !process.env.QUERY_END_DATE) {
    throw new Error(
        "QUERY_START_DATE and QUERY_END_DATE environment variables are required.",
    );
}

const cloudWatchQuery = new CloudWatchQuery(new CloudWatchLogsClient({}), {
    logGroupNames: ["/workflows/cloudwatch-logs/large-query"],
    dateRange: [
        new Date(Number.parseInt(process.env.QUERY_START_DATE)),
```

```

        new Date(Number.parseInt(process.env.QUERY_END_DATE)),
    ],
});

await cloudWatchQuery.run();

console.log(
    `Queries finished in ${cloudWatchQuery.secondsElapsed} seconds.\nTotal logs
    found: ${cloudWatchQuery.results.length}`,
);

```

Dies ist eine Klasse, die Abfragen bei Bedarf in mehrere Schritte aufteilt.

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
    StartQueryCommand,
    GetQueryResultsCommand,
} from "@aws-sdk/client-cloudwatch-logs";
import { splitDateRange } from "@aws-doc-sdk-examples/lib/utils/util-date.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

class DateOutOfBoundsError extends Error {}

export class CloudWatchQuery {
    /**
     * Run a query for all CloudWatch Logs within a certain date range.
     * CloudWatch logs return a max of 10,000 results. This class
     * performs a binary search across all of the logs in the provided
     * date range if a query returns the maximum number of results.
     *
     * @param {import('@aws-sdk/client-cloudwatch-logs').CloudWatchLogsClient}
    client
     * @param {{ logGroupNames: string[], dateRange: [Date, Date], queryConfig:
    { limit: number } }} config
     */
    constructor(client, { logGroupNames, dateRange, queryConfig }) {
        this.client = client;
    }
    /**
     * All log groups are queried.
     */
    this.logGroupNames = logGroupNames;

```

```

/**
 * The inclusive date range that is queried.
 */
this.dateRange = dateRange;

/**
 * CloudWatch Logs never returns more than 10,000 logs.
 */
this.limit = queryConfig?.limit ?? 10000;

/**
 * @type {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]}
 */
this.results = [];
}

/**
 * Run the query.
 */
async run() {
  this.secondsElapsed = 0;
  const start = new Date();
  this.results = await this._largeQuery(this.dateRange);
  const end = new Date();
  this.secondsElapsed = (end - start) / 1000;
  return this.results;
}

/**
 * Recursively query for logs.
 * @param {[Date, Date]} dateRange
 * @returns {Promise<import("@aws-sdk/client-cloudwatch-logs").ResultField[
[]>}
 */
async _largeQuery(dateRange) {
  const logs = await this._query(dateRange, this.limit);

  console.log(
    `Query date range: ${dateRange
      .map((d) => d.toISOString())
      .join(" to ")}. Found ${logs.length} logs.`
  );
}

```

```

    if (logs.length < this.limit) {
      return logs;
    }

    const lastLogDate = this._getLastLogDate(logs);
    const offsetLastLogDate = new Date(lastLogDate);
    offsetLastLogDate.setMilliseconds(lastLogDate.getMilliseconds() + 1);
    const subDateRange = [offsetLastLogDate, dateRange[1]];
    const [r1, r2] = splitDateRange(subDateRange);
    const results = await Promise.all([
      this._largeQuery(r1),
      this._largeQuery(r2),
    ]);
    return [logs, ...results].flat();
  }

  /**
   * Find the most recent log in a list of logs.
   * @param {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]} logs
   */
  _getLastLogDate(logs) {
    const timestamps = logs
      .map(
        (log) =>
          log.find((fieldMeta) => fieldMeta.field === "@timestamp")?.value,
      )
      .filter((t) => !!t)
      .map((t) => `${t}Z`)
      .sort();

    if (!timestamps.length) {
      throw new Error("No timestamp found in logs.");
    }

    return new Date(timestamps[timestamps.length - 1]);
  }

  /**
   * Simple wrapper for the GetQueryResultsCommand.
   * @param {string} queryId
   */
  _getQueryResults(queryId) {
    return this.client.send(new GetQueryResultsCommand({ queryId }));
  }

```



```

/**
 * Starts a query and waits for it to complete.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 */
async _query(dateRange, maxLogs) {
  try {
    const { queryId } = await this._startQuery(dateRange, maxLogs);
    const { results } = await this._waitUntilQueryDone(queryId);
    return results ?? [];
  } catch (err) {
    /**
     * This error is thrown when StartQuery returns an error indicating
     * that the query's start or end date occur before the log group was
     * created.
     */
    if (err instanceof DateOutOfBoundsError) {
      return [];
    }
    throw err;
  }
}

/**
 * Wrapper for the StartQueryCommand. Uses a static query string
 * for consistency.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 * @returns {Promise<{ queryId: string }>}
 */
async _startQuery([startDate, endDate], maxLogs = 10000) {
  try {
    return await this.client.send(
      new StartQueryCommand({
        logGroupNames: this.logGroupNames,
        queryString: "fields @timestamp, @message | sort @timestamp asc",
        startTime: startDate.valueOf(),
        endTime: endDate.valueOf(),
        limit: maxLogs,
      }),
    );
  } catch (err) {
    /** @type {string} */

```

```
    const message = err.message;
    if (message.startsWith("Query's end date and time")) {
      // This error indicates that the query's start or end date occur
      // before the log group was created.
      throw new DateOutOfBoundsError(message);
    }

    throw err;
  }
}

/**
 * Call GetQueryResultsCommand until the query is done.
 * @param {string} queryId
 */
_waitUntilQueryDone(queryId) {
  const getResults = async () => {
    const results = await this._getQueryResults(queryId);
    const queryDone = [
      "Complete",
      "Failed",
      "Cancelled",
      "Timeout",
      "Unknown",
    ].includes(results.status);

    return { queryDone, results };
  };

  return retry(
    { intervalInMs: 1000, maxRetries: 60, quiet: true },
    async () => {
      const { queryDone, results } = await getResults();
      if (!queryDone) {
        throw new Error("Query not done.");
      }

      return results;
    },
  );
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
- [GetQueryResults](#)
- [StartQuery](#)

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Diese Datei ruft ein Beispielm modul für die Verwaltung von CloudWatch Abfragen mit mehr als 10.000 Ergebnissen auf.

```
import logging
import os
import sys

import boto3
from botocore.config import Config

from cloudwatch_query import CloudWatchQuery
from date_utilities import DateUtilities

# Configure logging at the module level.
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(filename)s:%(lineno)d - %(message)s",
)

DEFAULT_QUERY_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"

class CloudWatchLogsQueryRunner:
    def __init__(self):
        """
```

```

        Initializes the CloudWatchLogsQueryRunner class by setting up date
        utilities
        and creating a CloudWatch Logs client with retry configuration.
        """
        self.date_utilities = DateUtilities()
        self.cloudwatch_logs_client = self.create_cloudwatch_logs_client()

    def create_cloudwatch_logs_client(self):
        """
        Creates and returns a CloudWatch Logs client with a specified retry
        configuration.

        :return: A CloudWatch Logs client instance.
        :rtype: boto3.client
        """
        try:
            return boto3.client("logs", config=Config(retries={"max_attempts":
10}))
        except Exception as e:
            logging.error(f"Failed to create CloudWatch Logs client: {e}")
            sys.exit(1)

    def fetch_environment_variables(self):
        """
        Fetches and validates required environment variables for query start and
        end dates.
        Fetches the environment variable for log group, returning the default
        value if it
        does not exist.

        :return: Tuple of query start date and end date as integers and the log
        group.
        :rtype: tuple
        :raises SystemExit: If required environment variables are missing or
        invalid.
        """
        try:
            query_start_date = int(os.environ["QUERY_START_DATE"])
            query_end_date = int(os.environ["QUERY_END_DATE"])
        except KeyError:
            logging.error(
                "Both QUERY_START_DATE and QUERY_END_DATE environment variables
                are required."
            )

```

```

        sys.exit(1)
    except ValueError as e:
        logging.error(f"Error parsing date environment variables: {e}")
        sys.exit(1)

    try:
        log_group = os.environ["QUERY_LOG_GROUP"]
    except KeyError:
        logging.warning("No QUERY_LOG_GROUP environment variable, using
default value")
        log_group = DEFAULT_QUERY_LOG_GROUP

    return query_start_date, query_end_date, log_group

def convert_dates_to_iso8601(self, start_date, end_date):
    """
    Converts UNIX timestamp dates to ISO 8601 format using DateUtilities.

    :param start_date: The start date in UNIX timestamp.
    :type start_date: int
    :param end_date: The end date in UNIX timestamp.
    :type end_date: int
    :return: Start and end dates in ISO 8601 format.
    :rtype: tuple
    """
    start_date_iso8601 =
self.date_utilities.convert_unix_timestamp_to_iso8601(
    start_date
)
    end_date_iso8601 = self.date_utilities.convert_unix_timestamp_to_iso8601(
    end_date
)
    return start_date_iso8601, end_date_iso8601

def execute_query(
    self,
    start_date_iso8601,
    end_date_iso8601,
    log_group="/workflows/cloudwatch-logs/large-query",
    query="fields @timestamp, @message | sort @timestamp asc"
):
    """
    Creates a CloudWatchQuery instance and executes the query with provided
date range.

```

```

:param start_date_iso8601: The start date in ISO 8601 format.
:type start_date_iso8601: str
:param end_date_iso8601: The end date in ISO 8601 format.
:type end_date_iso8601: str
:param log_group: Log group to search: "/workflows/cloudwatch-logs/large-
query"
:type log_group: str
:param query: Query string to pass to the CloudWatchQuery instance
:type query: str
"""
cloudwatch_query = CloudWatchQuery(
    log_group=log_group,
    query_string=query
)
cloudwatch_query.query_logs((start_date_iso8601, end_date_iso8601))
logging.info("Query executed successfully.")
logging.info(
    f"Queries completed in {cloudwatch_query.query_duration} seconds.
Total logs found: {len(cloudwatch_query.query_results)}"
)

def main():
    """
    Main function to start a recursive CloudWatch logs query.
    Fetches required environment variables, converts dates, and executes the
    query.
    """
    logging.info("Starting a recursive CloudWatch logs query...")
    runner = CloudWatchLogsQueryRunner()
    query_start_date, query_end_date, log_group =
runner.fetch_environment_variables()
    start_date_iso8601 = DateUtilities.convert_unix_timestamp_to_iso8601(
        query_start_date
    )
    end_date_iso8601 =
DateUtilities.convert_unix_timestamp_to_iso8601(query_end_date)
    runner.execute_query(start_date_iso8601, end_date_iso8601,
log_group=log_group)

if __name__ == "__main__":
    main()

```

Dieses Modul verarbeitet CloudWatch Abfragen mit mehr als 10.000 Ergebnissen.

```
import logging
import time
from datetime import datetime
import threading
import boto3

from date_utilities import DateUtilities

DEFAULT_QUERY = "fields @timestamp, @message | sort @timestamp asc"
DEFAULT_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"

class DateOutOfBoundsError(Exception):
    """Exception raised when the date range for a query is out of bounds."""

    pass

class CloudWatchQuery:
    """
    A class to query AWS CloudWatch logs within a specified date range.

    :vartype date_range: tuple
    :ivar limit: Maximum number of log entries to return.
    :vartype limit: int
    :log_group str: Name of the log group to query
    :query_string str: query
    """

    def __init__(self, log_group: str = DEFAULT_LOG_GROUP, query_string:
str=DEFAULT_QUERY) -> None:
        self.lock = threading.Lock()
        self.log_group = log_group
        self.query_string = query_string
        self.query_results = []
        self.query_duration = None
        self.datetime_format = "%Y-%m-%d %H:%M:%S.%f"
        self.date_utilities = DateUtilities()
        self.limit = 10000
```

```

def query_logs(self, date_range):
    """
    Executes a CloudWatch logs query for a specified date range and
    calculates the execution time of the query.

    :return: A batch of logs retrieved from the CloudWatch logs query.
    :rtype: list
    """
    start_time = datetime.now()

    start_date, end_date = self.date_utilities.normalize_date_range_format(
        date_range, from_format="unix_timestamp", to_format="datetime"
    )

    logging.info(
        f"Original query:"
        f"\n      START:      {start_date}"
        f"\n      END:        {end_date}"
        f"\n      LOG GROUP: {self.log_group}"
    )
    self.recursive_query((start_date, end_date))
    end_time = datetime.now()
    self.query_duration = (end_time - start_time).total_seconds()

def recursive_query(self, date_range):
    """
    Processes logs within a given date range, fetching batches of logs
    recursively if necessary.

    :param date_range: The date range to fetch logs for, specified as a tuple
    (start_timestamp, end_timestamp).
    :type date_range: tuple
    :return: None if the recursive fetching is continued or stops when the
    final batch of logs is processed.
        Although it doesn't explicitly return the query results, this
    method accumulates all fetched logs
        in the `self.query_results` attribute.
    :rtype: None
    """
    batch_of_logs = self.perform_query(date_range)
    # Add the batch to the accumulated logs
    with self.lock:
        self.query_results.extend(batch_of_logs)
    if len(batch_of_logs) == self.limit:

```



```

        logging.info(f"Fetched {self.limit}, checking for more...")
        most_recent_log = self.find_most_recent_log(batch_of_logs)
        most_recent_log_timestamp = next(
            item["value"]
            for item in most_recent_log
            if item["field"] == "@timestamp"
        )
        new_range = (most_recent_log_timestamp, date_range[1])
        midpoint = self.date_utilities.find_middle_time(new_range)

        first_half_thread = threading.Thread(
            target=self.recursive_query,
            args=((most_recent_log_timestamp, midpoint),),
        )
        second_half_thread = threading.Thread(
            target=self.recursive_query, args=((midpoint, date_range[1]),)
        )

        first_half_thread.start()
        second_half_thread.start()

        first_half_thread.join()
        second_half_thread.join()

def find_most_recent_log(self, logs):
    """
    Search a list of log items and return most recent log entry.
    :param logs: A list of logs to analyze.
    :return: log
    :type :return List containing log item details
    """
    most_recent_log = None
    most_recent_date = "1970-01-01 00:00:00.000"

    for log in logs:
        for item in log:
            if item["field"] == "@timestamp":
                logging.debug(f"Compared: {item['value']} to
{most_recent_date}")
                if (
                    self.date_utilities.compare_dates(
                        item["value"], most_recent_date
                    )
                    == item["value"]

```

```

        ):
            logging.debug(f"New most recent: {item['value']}")
            most_recent_date = item["value"]
            most_recent_log = log
        logging.info(f"Most recent log date of batch: {most_recent_date}")
        return most_recent_log

def perform_query(self, date_range):
    """
    Performs the actual CloudWatch log query.

    :param date_range: A tuple representing the start and end datetime for
    the query.
    :type date_range: tuple
    :return: A list containing the query results.
    :rtype: list
    """
    client = boto3.client("logs")
    try:
        try:
            start_time = round(
self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(
self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=self.limit,
            )
            query_id = response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")
        while True:
            time.sleep(1)
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",

```

```

        "Cancelled",
        "Timeout",
        "Unknown",
    ]:
        return results.get("results", [])
    except DateOutOfBoundsError:
        return []

def _initiate_query(self, client, date_range, max_logs):
    """
    Initiates the CloudWatch logs query.

    :param date_range: A tuple representing the start and end datetime for
the query.
    :type date_range: tuple
    :param max_logs: The maximum number of logs to retrieve.
    :type max_logs: int
    :return: The query ID as a string.
    :rtype: str
    """
    try:
        start_time = round(
self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
        )
        end_time = round(
self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
        )
        response = client.start_query(
            logGroupName=self.log_group,
            startTime=start_time,
            endTime=end_time,
            queryString=self.query_string,
            limit=max_logs,
        )
        return response["queryId"]
    except client.exceptions.ResourceNotFoundException as e:
        raise DateOutOfBoundsError(f"Resource not found: {e}")

def _wait_for_query_results(self, client, query_id):
    """
    Waits for the query to complete and retrieves the results.

```

```
:param query_id: The ID of the initiated query.
:type query_id: str
:return: A list containing the results of the query.
:rtype: list
"""
while True:
    time.sleep(1)
    results = client.get_query_results(queryId=query_id)
    if results["status"] in [
        "Complete",
        "Failed",
        "Cancelled",
        "Timeout",
        "Unknown",
    ]:
        return results.get("results", [])
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
 - [GetQueryResults](#)
 - [StartQuery](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine AWS Lambda Funktion erstellt wird, die durch ein von Amazon EventBridge geplantes Ereignis aufgerufen wird.

Java

SDK für Java 2.x

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge, dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel

erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge, dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der JavaScript Lambda-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

Dieses Beispiel ist auch verfügbar im [AWS SDK für JavaScript Entwicklerhandbuch für v3](#).

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda

- Amazon SNS

Python

SDK für Python (Boto3)

Dieses Beispiel zeigt, wie eine AWS Lambda Funktion als Ziel einer geplanten EventBridge Amazon-Veranstaltung registriert wird. Der Lambda-Handler schreibt eine freundliche Nachricht und die vollständigen Ereignisdaten für den späteren Abruf in Amazon CloudWatch Logs.

- Stellt eine Lambda-Funktion bereit.
- Erzeugt ein EventBridge geplantes Ereignis und macht die Lambda-Funktion zum Ziel.
- Erteilt die Erlaubnis, die EventBridge Lambda-Funktion aufrufen zu lassen.
- Druckt die neuesten Daten aus CloudWatch Logs, um das Ergebnis der geplanten Aufrufe anzuzeigen.
- Bereinigt alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten auf ansehen. [GitHub](#) Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Verwenden von CloudWatch Logs mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verschlüsseln Sie Nachschlagetabellen in CloudWatch Protokollen mit AWS Key Management Service

Die Daten der Nachschlagetabellen werden in CloudWatch Protokollen immer verschlüsselt. Standardmäßig verwendet CloudWatch Logs serverseitige Verschlüsselung mit 256-Bit-AES-GCM (Advanced Encryption Standard Galois/Counter Mode), um ruhende Daten in der Nachschlagetabelle zu verschlüsseln. Als Alternative können Sie AWS Key Management Service für diese Verschlüsselung verwenden. Wenn Sie dies tun, erfolgt die Verschlüsselung mithilfe eines Schlüssels. AWS KMS Die Verschlüsselung mit AWS KMS wird auf der Ebene der Nachschlagetabelle aktiviert, indem ein KMS-Schlüssel einer Nachschlagetabelle zugeordnet wird, entweder wenn Sie die Nachschlagetabelle erstellen oder wenn Sie sie aktualisieren.

Important

CloudWatch Logs unterstützt nur symmetrische KMS-Schlüssel. Verwenden Sie keinen asymmetrischen Schlüssel, um die Daten in Ihren Nachschlagetabellen zu verschlüsseln. Weitere Informationen finden Sie unter [Verwenden von symmetrischen und asymmetrischen Schlüsseln](#).

Nachdem Sie einer Nachschlagetabelle einen KMS-Schlüssel zugeordnet haben, werden alle in der Nachschlagetabelle gespeicherten Daten mit diesem Schlüssel verschlüsselt. CloudWatch Logs entschlüsselt diese Daten, wann immer sie angefordert werden. CloudWatch Protokolle müssen über Berechtigungen für den KMS-Schlüssel verfügen, wenn verschlüsselte Daten angefordert werden.

Wenn Sie später die Zuordnung eines KMS-Schlüssels zu einer Nachschlagetabelle aufheben, verschlüsselt CloudWatch Logs die Daten mit der CloudWatch Standardverschlüsselungsmethode Logs. Wenn der Schlüssel jedoch deaktiviert oder gelöscht wird, bevor Sie die Zuordnung aufheben, kann CloudWatch Logs die Daten, die mit diesem Schlüssel verschlüsselt wurden, nicht lesen.

Allgemeine Informationen darüber, wie CloudWatch Logs AWS KMS Protokolldaten verschlüsselt, finden Sie unter. [Verschlüsseln Sie Protokolldaten in CloudWatch Logs mit AWS Key Management Service](#)

Wie CloudWatch Logs AWS KMS für Lookup-Tabellen verwendet

CloudWatch Logs verwendet die AWS KMS Umschlagverschlüsselung, um die Daten der Nachschlagetabellen zu schützen. Wenn Sie einer Nachschlagetabelle einen KMS-Schlüssel zuordnen, sendet CloudWatch Logs eine `GenerateDataKey` Anfrage an AWS KMS. AWS KMS generiert einen eindeutigen Datenverschlüsselungsschlüssel (DEK) und gibt sowohl eine Klartext-Kopie als auch eine verschlüsselte Kopie des DEK zurück. CloudWatch Logs verwendet das Klartext-DEK, um die Daten der Nachschlagetabelle zu verschlüsseln, und speichert dann das verschlüsselte DEK zusammen mit den verschlüsselten Daten. Das Klartext-DEK wird nicht gespeichert und nach der Verwendung aus dem Speicher gelöscht.

Wenn CloudWatch Logs die Daten der Nachschlagetabelle lesen muss, sendet es eine `Decrypt` Anfrage AWS KMS mit dem verschlüsselten DEK an. AWS KMS entschlüsselt das DEK und gibt das Klartext-DEK an CloudWatch Logs zurück, wo es dann zur Entschlüsselung der Daten der Nachschlagetabelle verwendet wird.

CloudWatch Logs verwendet den folgenden Verschlüsselungskontext für Anfragen an: AWS KMS

```
{
  "aws:logs:arn": "arn:aws:logs:region:account-id:lookup-table:lookup-table-name"
}
```

Sie können diesen Verschlüsselungskontext in IAM-Richtlinien und AWS KMS Schlüsselrichtlinien verwenden, um den Zugriff auf den KMS-Schlüssel zu steuern. Weitere Informationen finden Sie unter [AWS KMS Schlüssel und Verschlüsselungskontext](#).

Erforderliche -Berechtigungen

Um die AWS KMS Verschlüsselung mit Nachschlagetabellen verwenden zu können, muss der IAM-Prinzipal über die folgenden AWS KMS Berechtigungen für den KMS-Schlüssel verfügen:

- `kms:Decrypt`
- `kms:GenerateDataKey`

Die `kms:Decrypt` Berechtigung ist erforderlich, wenn Sie eine Nachschlagetabelle aufrufen `GetLookupTable`, die mit einem KMS-Schlüssel verschlüsselt ist, sodass CloudWatch Logs die Daten in Ihrem Namen entschlüsseln kann. Die `kms:Decrypt` Berechtigung ist auch für

den Schlüssel (den KMS-Schlüssel, der zum Verschlüsseln der Nachschlagetabelle verwendet wird) erforderlich, wenn `StartQuery` mit einer Abfrage aufgerufen wird, die den `lookup` Befehl für eine verschlüsselte Nachschlagetabelle verwendet. Die `kms:GenerateDataKey` Berechtigung ist beim Aufrufen `CreateLookupTable` oder `UpdateLookupTable` mit einem KMS-Schlüssel erforderlich, sodass CloudWatch Logs einen Datenverschlüsselungsschlüssel zum Verschlüsseln der Daten der Nachschlagetabelle generieren kann.

Darüber hinaus muss der CloudWatch Logs-Dienst über die Berechtigung zur Verwendung des KMS-Schlüssels verfügen. Sie gewähren diese Berechtigungen, indem Sie der KMS-Schlüsselrichtlinie eine Richtlinienanweisung hinzufügen, wie im folgenden Abschnitt beschrieben.

Schritt 1: Erstellen Sie einen AWS KMS Schlüssel

Verwenden Sie den folgenden Befehl [create-key](#), um einen symmetrischen KMS-Schlüssel zu erstellen:

```
aws kms create-key
```

Die Ausgabe enthält die Schlüssel-ID und den Amazon-Ressourcennamen (ARN) des Schlüssels. Das Folgende ist eine Beispielausgabe:

```
{
  "KeyMetadata": {
    "Origin": "AWS_KMS",
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "Description": "",
    "KeyManager": "CUSTOMER",
    "Enabled": true,
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "KeyState": "Enabled",
    "CreationDate": 1478910250.94,
    "Arn": "arn:aws:kms:us-west-2:123456789012:key/6f815f63-e628-448c-8251-
e40cb0d29f59",
    "AWSAccountId": "123456789012",
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ]
  }
}
```

Notieren Sie sich den Schlüssel-ARN. Sie benötigen ihn in den folgenden Schritten.

Schritt 2: Festlegen von Berechtigungen auf dem KMS-Schlüssel

Standardmäßig sind alle AWS KMS Schlüssel privat. Nur der Ressourcenbesitzer kann mit ihnen Daten verschlüsseln und entschlüsseln. Sie müssen dem Prinzipal des CloudWatch Logs-Dienstes die Berechtigung zur Verwendung des Schlüssels und auch der aufrufenden Rolle die Berechtigung zur Verwendung des Schlüssels erteilen.

Speichern Sie zunächst die Standardrichtlinie für Ihren KMS-Schlüssel `policy.json` mit dem folgenden [get-key-policy](#) Befehl:

```
aws kms get-key-policy --key-id key-id --policy-name default --output text > ./policy.json
```

Öffnen Sie die `policy.json` Datei in einem Texteditor und fügen Sie die folgende Anweisung hinzu, um dem Prinzipal des CloudWatch Logs-Dienstes die Berechtigung zur Verwendung des Schlüssels zu erteilen. In diesem Beispiel wird ein Condition Abschnitt verwendet, der dem Verschlüsselungskontext entspricht, um den Schlüssel auf eine bestimmte Nachschlagetabelle zu beschränken.

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "logs.region.amazonaws.com"
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:region:account-id:lookup-table:lookup-table-name",
      "aws:SourceAccount": "account-id",
      "aws:SourceArn": "arn:aws:logs:region:account-id:lookup-table:lookup-table-name"
    }
  }
}
```

}

Als Nächstes fügen Sie der Rolle, die die CloudWatch Logs `CreateLookupTable` oder aufrufen soll, Berechtigungen hinzu `UpdateLookupTable` APIs. CloudWatch Logs wird verwendet `kms:ViaService`, um AWS KMS im Namen des Kunden Anrufe zu tätigen. Weitere Informationen finden Sie unter [kms: ViaService](#).

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::account-id:role/role-name"
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": [
        "logs.region.amazonaws.com"
      ]
    }
  }
}
```

Fügen Sie abschließend die aktualisierte Richtlinie mit dem folgenden [put-key-policy](#) Befehl hinzu:

```
aws kms put-key-policy --key-id key-id --policy-name default --policy file://
policy.json
```

Schritt 3: Verknüpfen Sie einen KMS-Schlüssel mit einer Nachschlagetabelle

Sie können einen KMS-Schlüssel einer Nachschlagetabelle zuordnen, wenn Sie ihn mithilfe der `CreateLookupTable` API erstellen, oder Sie können eine vorhandene Nachschlagetabelle mithilfe der `UpdateLookupTable` API aktualisieren. Beide APIs sind Teil der `AWSLogsConfigService`.

Um den KMS-Schlüssel bei der Erstellung einer Nachschlagetabelle zuzuordnen

Verwenden Sie die `CreateLookupTable` API und geben Sie den `kmsKeyArn` Parameter mit dem ARN Ihres KMS-Schlüssels an:

```
aws logs create-lookup-table \  
  --lookup-table-name my-lookup-table \  
  --kms-key-arn "arn:aws:kms:region:account-id:key/key-id"
```

Um den KMS-Schlüssel einer vorhandenen Nachschlagetabelle zuzuordnen

Verwenden Sie die `UpdateLookupTable` API und geben Sie den `kmsKeyArn` Parameter mit dem ARN Ihres KMS-Schlüssels an:

```
aws logs update-lookup-table \  
  --lookup-table-name my-lookup-table \  
  --kms-key-arn "arn:aws:kms:region:account-id:key/key-id"
```

Überlegungen

- Nachdem Sie einen KMS-Schlüssel einer Nachschlagetabelle zugeordnet oder die Verknüpfung aufgehoben haben, kann es bis zu fünf Minuten dauern, bis der Vorgang wirksam wird.
- Wenn Sie CloudWatch Logs den Zugriff auf einen zugehörigen Schlüssel entziehen oder einen zugehörigen KMS-Schlüssel löschen, können Ihre verschlüsselten Nachschlagetabellendaten in CloudWatch Logs nicht mehr abgerufen werden.
- Um die Schritte in diesem Thema ausführen zu können, benötigen Sie die folgenden Berechtigungen: `kms:CreateKey`, `kms:GetKeyPolicy`, `kms:PutKeyPolicy`, und die entsprechenden CloudWatch Logs-Berechtigungen, um `CreateLookupTable` oder `UpdateLookupTable` aufrufen zu können.

Sicherheit in Amazon CloudWatch Logs

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das [Modell der übergreifenden Verantwortlichkeit](#) beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- **Sicherheit der Cloud** — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der [AWS](#). Weitere Informationen zu den Compliance-Programmen, die für gelten WorkSpaces, finden Sie unter [AWS Services im Umfang nach Compliance-Programmen AWS](#).
- **Sicherheit in der Cloud** — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung von Amazon CloudWatch Logs anwenden können. Es zeigt Ihnen, wie Sie Amazon CloudWatch Logs konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere AWS Dienste nutzen können, die Ihnen bei der Überwachung und Sicherung Ihrer CloudWatch Logs-Ressourcen helfen.

Inhalt

- [Datenschutz in Amazon CloudWatch Logs](#)
- [Identitäts- und Zugriffsmanagement für Amazon CloudWatch Logs](#)
- [Konformitätsprüfung für Amazon CloudWatch Logs](#)
- [Resilienz in Amazon CloudWatch Logs](#)
- [Infrastruktursicherheit in Amazon CloudWatch Logs](#)
- [Verwenden von CloudWatch Protokollen mit VPC-Endpunkten der Schnittstelle](#)

Datenschutz in Amazon CloudWatch Logs

Note

Zusätzlich zu den folgenden Informationen zum allgemeinen Datenschutz in AWS können Sie mit CloudWatch Logs auch sensible Daten in Protokollereignissen schützen, indem Sie sie maskieren. Weitere Informationen finden Sie unter [Den Schutz vertraulicher Protokolldaten mit Maskierung unterstützen](#).

Das AWS [Modell](#) der gilt für den Datenschutz in Amazon CloudWatch Logs. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter [Häufig gestellte Fragen zum Datenschutz](#) . Informationen zum Datenschutz in Europa finden Sie im [General Data Protection Regulation \(GDPR\) Center](#).

Aus Datenschutzgründen empfehlen wir, dass Sie Ihre AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter [Arbeiten mit CloudTrail Pfaden](#) im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere

Informationen über verfügbare FIPS-Endpunkte finden Sie unter [Federal Information Processing Standard \(FIPS\) 140-3](#).

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit CloudWatch Protokollen oder anderen Methoden AWS-Services über die Konsole, API oder SDKs arbeiten. AWS CLI AWS Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Verschlüsselung im Ruhezustand

CloudWatch Logs schützt Daten im Ruhezustand mithilfe von Verschlüsselung. Alle Protokollgruppen sind verschlüsselt. Standardmäßig verwaltet der CloudWatch Logs-Dienst die serverseitige Verschlüsselung und verwendet serverseitige Verschlüsselung mit 256-Bit Advanced Encryption Standard Galois/Counter Mode (AES-GCM), um Protokolldaten im Ruhezustand zu verschlüsseln.

Wenn Sie die Schlüssel verwalten möchten, die zum Verschlüsseln und Entschlüsseln Ihrer Protokolle verwendet werden, verwenden Sie Schlüssel. AWS KMS Weitere Informationen finden Sie unter [Verschlüsseln Sie Protokolldaten in CloudWatch Logs mit AWS Key Management Service](#).

Verschlüsselung während der Übertragung

CloudWatch Logs verwendet eine Ende-zu-Ende-Verschlüsselung von Daten während der Übertragung. Der CloudWatch Logs-Dienst verwaltet die serverseitigen Verschlüsselungsschlüssel.

Identitäts- und Zugriffsmanagement für Amazon CloudWatch Logs

Für den Zugriff auf Amazon CloudWatch Logs sind Anmeldeinformationen erforderlich, mit denen Sie Ihre Anfragen authentifizieren AWS können. Diese Anmeldeinformationen müssen über Berechtigungen für den Zugriff auf AWS Ressourcen verfügen, z. B. zum Abrufen von CloudWatch Logs-Daten über Ihre Cloud-Ressourcen. In den folgenden Abschnitten erfahren Sie, wie Sie [AWS Identity and Access Management \(IAM\)](#) und CloudWatch Logs verwenden können, um Ihre Ressourcen zu schützen, indem Sie kontrollieren, wer darauf zugreifen kann:

- [Authentifizierung](#)

- [Zugriffskontrolle](#)

Authentifizierung

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter [Erstellen eines Berechtigungssatzes](#) im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter [Eine Rolle für einen externen Identitätsanbieter \(Verbund\) erstellen](#) im IAM-Benutzerhandbuch.

- IAM-Benutzer:

- Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter [Eine Rolle für einen IAM-Benutzer erstellen](#) im IAM-Benutzerhandbuch.
- (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter [Hinzufügen von Berechtigungen zu einem Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

Zugriffskontrolle

Sie können über gültige Anmeldeinformationen verfügen, um Ihre Anfragen zu authentifizieren, aber ohne die entsprechenden Berechtigungen können Sie keine CloudWatch Logs-Ressourcen erstellen oder darauf zugreifen. Sie müssen beispielsweise über Berechtigungen verfügen, um Protokoll-Streams, Protokollgruppen und so weiter zu erstellen.

In den folgenden Abschnitten wird beschrieben, wie Sie die Berechtigungen für CloudWatch Logs verwalten. Wir empfehlen Ihnen, zunächst die Übersicht zu lesen.

- [Überblick über die Verwaltung der Zugriffsberechtigungen für Ihre CloudWatch Logs-Ressourcen](#)
- [Verwenden von identitätsbasierten Richtlinien \(IAM-Richtlinien\) für Protokolle CloudWatch](#)
- [CloudWatch Referenz zu Protokollberechtigungen](#)

Überblick über die Verwaltung der Zugriffsberechtigungen für Ihre CloudWatch Logs-Ressourcen

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter [Erstellen eines Berechtigungssatzes](#) im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anleitung unter [Eine Rolle für einen externen Identitätsanbieter \(Verbund\) erstellen](#) im IAM-Benutzerhandbuch.

- IAM-Benutzer:

- Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Befolgen Sie die Anleitung unter [Eine Rolle für einen IAM-Benutzer erstellen](#) im IAM-Benutzerhandbuch.
- (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter [Hinzufügen von Berechtigungen zu einem Benutzer \(Konsole\)](#) im IAM-Benutzerhandbuch.

Themen

- [CloudWatch Protokolliert Ressourcen und Operationen](#)
- [Grundlegendes zum Eigentum an Ressourcen](#)
- [Verwaltung des Zugriffs auf -Ressourcen](#)
- [Festlegen der Richtlinienelemente: Aktionen, Effekte und Prinzipale](#)
- [Angaben von Bedingungen in einer Richtlinie](#)

CloudWatch Protokolliert Ressourcen und Operationen

In CloudWatch Logs sind die Hauptressourcen Log-Gruppen, Log-Streams und Ziele. CloudWatch Logs unterstützt keine Unterressourcen (andere Ressourcen zur Verwendung mit der primären Ressource).

Diesen Ressourcen und Unterressourcen sind eindeutige Amazon-Ressourcennamen (ARN) zugeordnet, wie in der folgenden Tabelle zu sehen ist.

Ressourcentyp	ARN-Format
Gruppe protokollieren	arn:aws:logs: ::log-group: <i>region account-id log_group_name</i>
Protokoll-Stream	arn:aws:logs: ::log-group <i>region</i> ::log-stream: <i>account-id log_group_name log-stream-name</i>
Ziel	arn:aws:logs: <i>region</i> ::Ziel: <i>account-id destination_name</i>

Weitere Informationen zu ARNs finden Sie unter [ARNs](#) im IAM-Benutzerhandbuch. Informationen zu CloudWatch Logs-ARNs finden Sie unter [Amazon Resource Names \(ARNs\)](#) in. Allgemeine Amazon Web Services-Referenz Ein Beispiel für eine Richtlinie, die CloudWatch Logs abdeckt, finden Sie unter. [Verwenden von identitätsbasierten Richtlinien \(IAM-Richtlinien\) für Protokolle CloudWatch](#)

CloudWatch Logs bietet eine Reihe von Vorgängen für die Arbeit mit den CloudWatch Logs-Ressourcen. Eine Liste der verfügbaren Operationen finden Sie unter [CloudWatch Referenz zu Protokollberechtigungen](#).

Grundlegendes zum Eigentum an Ressourcen

Das AWS Konto besitzt die Ressourcen, die im Konto erstellt wurden, unabhängig davon, wer die Ressourcen erstellt hat. Insbesondere ist der Ressourcenbesitzer das AWS Konto der [Prinzipalidentität](#) (d. h. das Root-Konto, ein Benutzer oder eine IAM-Rolle), das die Anfrage zur Ressourcenerstellung authentifiziert. Die Funktionsweise wird anhand der folgenden Beispiele deutlich:

- Wenn Sie die Root-Kontoanmeldeinformationen Ihres AWS Kontos verwenden, um eine Protokollgruppe zu erstellen, ist Ihr AWS Konto der Eigentümer der CloudWatch Logs-Ressource.
- Wenn Sie in Ihrem AWS Konto einen Benutzer erstellen und diesem Benutzer Berechtigungen zum Erstellen von CloudWatch Logs-Ressourcen gewähren, kann der Benutzer CloudWatch Logs-Ressourcen erstellen. Ihr AWS Konto, zu dem der Benutzer gehört, besitzt jedoch die CloudWatch Logs-Ressourcen.

- Wenn Sie in Ihrem AWS Konto eine IAM-Rolle mit Berechtigungen zum Erstellen von CloudWatch Logs-Ressourcen erstellen, kann jeder, der diese Rolle übernehmen kann, CloudWatch Logs-Ressourcen erstellen. Ihr AWS Konto, zu dem die Rolle gehört, besitzt die CloudWatch Logs-Ressourcen.

Verwaltung des Zugriffs auf -Ressourcen

Eine Berechtigungsrichtlinie beschreibt, wer Zugriff auf welche Objekte hat. Im folgenden Abschnitt werden die verfügbaren Optionen zum Erstellen von Berechtigungsrichtlinien erläutert.

Note

In diesem Abschnitt wird die Verwendung von IAM im Kontext von CloudWatch Protokollen beschrieben. Er enthält keine detaillierten Informationen über den IAM-Service. Eine umfassende IAM-Dokumentation finden Sie unter [Was ist IAM?](#) im IAM-Benutzerhandbuch. Informationen über die Syntax und Beschreibungen von IAM-Richtlinien finden Sie in der [IAM-Richtlinienreferenz](#) im IAM-Benutzerhandbuch.

Richtlinien, die mit einer IAM-Identität verknüpft sind, werden als identitätsbasierte Richtlinien (IAM-Richtlinien) bezeichnet, und Richtlinien, die einer Ressource zugeordnet sind, werden als ressourcenbasierte Richtlinien bezeichnet. CloudWatch Logs unterstützt identitätsbasierte Richtlinien und ressourcenbasierte Richtlinien für Ziele, die verwendet werden, um kontoübergreifende Abonnements zu ermöglichen. Weitere Informationen finden Sie unter [Kontoübergreifende, regionsübergreifende Abonnements](#).

Themen

- [Protokollgruppen-Berechtigungen und Contributor Insights](#)
- [Resource-based Richtlinien](#)

Protokollgruppen-Berechtigungen und Contributor Insights

Contributor Insights ist eine Funktion von CloudWatch , mit der Sie Daten aus Protokollgruppen analysieren und Zeitreihen erstellen können, in denen Daten von Mitwirkenden angezeigt werden. Sie können Metriken über die Top-N-Contributors, die Gesamtzahl der eindeutigen Contributors und deren Nutzung anzeigen. Weitere Informationen finden Sie unter [Verwenden von Contributor Insights zur Datenanalyse](#). High-Cardinality

Wenn Sie einem Benutzer die `cloudwatch:GetInsightRuleReport` Berechtigungen `cloudwatch:PutInsightRule` und gewähren, kann dieser Benutzer eine Regel erstellen, die jede Protokollgruppe in CloudWatch Logs auswertet, und sich dann die Ergebnisse ansehen. Die Ergebnisse können Contributor-Daten für diese Protokollgruppen enthalten. Gewähren Sie diese Berechtigungen nur Benutzern, die diese Daten anzeigen können sollten.

Resource-based Richtlinien

CloudWatch Logs unterstützt ressourcenbasierte Richtlinien für Ziele, mit denen Sie kontoübergreifende Abonnements aktivieren können. Weitere Informationen finden Sie unter [Schritt 1: Erstellen eines Ziels](#). Ziele können mithilfe der [PutDestination](#) API erstellt werden, und Sie können dem Ziel mithilfe der [PutDestinationPolicy](#) API eine Ressourcenrichtlinie hinzufügen. Beim folgenden Beispiel wird es einem anderen AWS -Konto mit der Konto-ID 111122223333 ermöglicht, dass seine Protokollgruppen das Ziel `arn:aws:logs:us-east-1:123456789012:destination:testDestination` abonnieren.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "logs:PutSubscriptionFilter",
      "Resource": "arn:aws:logs:us-east-1:123456789012:destination:testDestination"
    }
  ]
}
```

Festlegen der Richtlinienelemente: Aktionen, Effekte und Prinzipale

Für jede CloudWatch Logs-Ressource definiert der Dienst eine Reihe von API-Vorgängen. Um Berechtigungen für diese API-Operationen zu gewähren, definiert CloudWatch Logs eine Reihe von Aktionen, die Sie in einer Richtlinie angeben können. Einige API-Operationen erfordern

möglicherweise Berechtigungen für mehr als eine Aktion, um die API-Operation auszuführen. Weitere Informationen zu Ressourcen und API-Operationen finden Sie unter [CloudWatch Protokolliert Ressourcen und Operationen](#) und [CloudWatch Referenz zu Protokollberechtigungen](#).

Grundlegende Richtlinienelemente:

- **Ressource** – Sie verwenden einen Amazon-Ressourcennamen (ARN), um die Ressource, für welche die Richtlinie gilt, zu identifizieren. Weitere Informationen finden Sie unter [CloudWatch Protokolliert Ressourcen und Operationen](#).
- **Aktion** – Mit Aktionsschlüsselwörtern geben Sie die Ressourcenoperationen an, die Sie zulassen oder verweigern möchten. Die `logs.DescribeLogGroups`-Berechtigung erteilt dem Benutzer zum Beispiel Berechtigungen zum Ausführen der `DescribeLogGroups`-Operation.
- **Effekt** – Die von Ihnen festgelegte Auswirkung (entweder Zugriffserlaubnis oder Zugriffsverweigerung), wenn ein Benutzer die jeweilige Aktion anfordert. Wenn Sie den Zugriff auf eine Ressource nicht ausdrücklich gestatten („Allow“), wird er automatisch verweigert. Sie können den Zugriff auf eine Ressource auch explizit verweigern. So können Sie sicherstellen, dass Benutzer nicht darauf zugreifen können, auch wenn der Zugriff durch eine andere Richtlinie gestattet wird.
- **Prinzipal** – In identitätsbasierten Richtlinien (IAM-Richtlinien) ist der Benutzer, dem die Richtlinie zugewiesen ist, automatisch der Prinzipal. Bei ressourcenbasierten Richtlinien geben Sie den Benutzer, das Konto, den Dienst oder die andere Entität an, für die Sie Berechtigungen erhalten möchten (gilt nur für ressourcenbasierte Richtlinien). CloudWatch Logs unterstützt ressourcenbasierte Richtlinien für Ziele.

Weitere Informationen zur Syntax und zu Beschreibungen von IAM-Richtlinien finden Sie in der [AWS -IAM-Richtlinienreferenz](#) im IAM-Benutzerhandbuch.

Eine Tabelle mit allen CloudWatch Logs-API-Aktionen und den Ressourcen, für die sie gelten, finden Sie unter [CloudWatch Referenz zu Protokollberechtigungen](#)

Angeben von Bedingungen in einer Richtlinie

Beim Erteilen von Berechtigungen können Sie mithilfe der Sprache der Zugriffsrichtlinie die Bedingungen angeben, wann die Richtlinie wirksam werden soll. Beispielsweise kann festgelegt werden, dass eine Richtlinie erst ab einem bestimmten Datum gilt. Weitere Informationen zum Angeben von Bedingungen in einer Richtlinienyntax finden Sie im Thema [Bedingung](#) im IAM Benutzerhandbuch.

Bedingungen werden mithilfe vordefinierter Bedingungsschlüssel formuliert. Eine Liste der von den einzelnen AWS Diensten unterstützten Kontextschlüssel und eine Liste der allgemeinen AWS Richtlinienchlüssel finden Sie unter [Aktionen, Ressourcen und Bedingungsschlüssel für AWS Dienste](#) und [AWS globale Bedingungskontextschlüssel](#).

Note

Sie können Tags verwenden, um den Zugriff auf CloudWatch Logs-Ressourcen, einschließlich Protokollgruppen und -ziele, zu kontrollieren. Der Zugriff auf Protokollstreams wird aufgrund der hierarchischen Beziehung zwischen Protokollgruppen und Protokollstreams auf der Ebene der Protokollgruppen gesteuert. Informationen über die Verwendung von Tags zum Steuern des Zugriffs finden Sie unter [Steuern des Zugriffs auf Amazon-Web-Services-Ressourcen mithilfe von Tags](#).

Verwenden von identitätsbasierten Richtlinien (IAM-Richtlinien) für Protokolle CloudWatch

In diesem Thema finden Sie Beispiele für identitätsbasierte Richtlinien, in denen ein Kontoadministrator den IAM-Identitäten (Benutzer, Gruppen und Rollen) Berechtigungsrichtlinien anfügen kann.

Important

Wir empfehlen Ihnen, zunächst die einführenden Themen zu lesen, in denen die grundlegenden Konzepte und Optionen erläutert werden, die Ihnen zur Verwaltung des Zugriffs auf Ihre CloudWatch Logs-Ressourcen zur Verfügung stehen. Weitere Informationen finden Sie unter [Überblick über die Verwaltung der Zugriffsberechtigungen für Ihre CloudWatch Logs-Ressourcen](#).

In diesem Thema wird Folgendes behandelt:

- [Für die Verwendung der CloudWatch Konsole sind Berechtigungen erforderlich](#)
- [AWS verwaltete \(vordefinierte\) Richtlinien für CloudWatch Protokolle](#)
- [Beispiele für vom Kunden verwaltete Richtlinien](#)

Nachstehend finden Sie ein Beispiel für eine Berechtigungsrichtlinie:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:*"
      ]
    }
  ]
}
```

In dieser Richtlinie gibt es eine Anweisung, die Berechtigungen erteilt, um Protokollgruppen und Protokoll-Streams zu erstellen, Protokollereignisse hochzuladen und Details zu Protokoll-Streams aufzuführen.

Das Platzhalterzeichen (*) im Resource Wert gewährt die Erlaubnis für die aufgelisteten Aktionen für jede CloudWatch Logs-Ressource.

- Um die Berechtigungen auf bestimmte Protokollgruppenaktionen zu beschränken (z. B. `CreateLogGroup`, `DescribeLogStreams`), ersetzen Sie den Platzhalter durch den Protokollgruppen-ARN— `arn:aws:logs:us-west-2:123456789012:log-group:MyLogGroup`
- Um die Berechtigungen auf bestimmte Logstream-Aktionen zu beschränken (z. B. `CreateLogStream`, `PutLogEvents`), ersetzen Sie den Platzhalter durch den Log-Stream ARN — `arn:aws:logs:us-west-2:123456789012:log-group:MyLogGroup:*` (entspricht allen Streams) oder `arn:aws:logs:us-west-2:123456789012:log-group:MyLogGroup:log-stream:MyStream` (bestimmter Stream)

In der [Referenz zur Serviceautorisierung](#) finden Sie Informationen darüber, welchen Ressourcentyp jede API benötigt.

Für die Verwendung der CloudWatch Konsole sind Berechtigungen erforderlich

Damit ein Benutzer mit CloudWatch Logs in der CloudWatch Konsole arbeiten kann, muss er über Mindestberechtigungen verfügen, die es dem Benutzer ermöglichen, andere AWS Ressourcen in seinem AWS Konto zu beschreiben. Um CloudWatch Logs in der CloudWatch Konsole verwenden zu können, benötigen Sie Berechtigungen für die folgenden Dienste:

- CloudWatch
- CloudWatch Logs
- OpenSearch Bedienung
- IAM
- Kinesis
- Lambda
- Amazon S3

Wenn Sie eine IAM-Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Benutzer mit dieser IAM-Richtlinie. Um sicherzustellen, dass diese Benutzer die CloudWatch Konsole weiterhin verwenden können, fügen Sie dem Benutzer auch die `CloudWatchReadOnlyAccess` verwaltete Richtlinie bei, wie unter [beschrieben](#) [AWS verwaltete \(vordefinierte\) Richtlinien für CloudWatch Protokolle](#).

Sie müssen Benutzern, die nur die Logs-API AWS CLI oder die CloudWatch Logs-API aufrufen, keine Mindestberechtigungen für die Konsole gewähren.

Für Benutzer, die die CloudWatch Konsole nicht zur Verwaltung von Protokollabonnements verwenden, sind die folgenden Berechtigungen erforderlich, um mit der Konsole zu arbeiten:

- Cloudwatch: GetMetricData
- Cloudwatch: ListMetrics
- Logs: CancelExportTask
- Protokolle: CreateExportTask
- Protokolle: CreateLogGroup

- Protokolle: CreateLogStream
- Protokolle: DeleteLogGroup
- Protokolle: DeleteLogStream
- Protokolle: DeleteMetricFilter
- Protokolle: DeleteQueryDefinition
- Protokolle: DeleteRetentionPolicy
- Protokolle: DeleteSubscriptionFilter
- Protokolle: DescribeExportTasks
- Protokolle: DescribeLogGroups
- Protokolle: DescribeLogStreams
- Protokolle: DescribeMetricFilters
- Protokolle: DescribeQueryDefinitions
- Protokolle: DescribeQueries
- Protokolle: DescribeSubscriptionFilters
- Protokolle: FilterLogEvents
- Protokolle: GetLogEvents
- Protokolle: GetLogGroupFields
- Protokolle: GetLogRecord
- Protokolle: GetQueryResults
- Protokolle: PutMetricFilter
- Protokolle: PutQueryDefinition
- Protokolle: PutRetentionPolicy
- Protokolle: StartQuery
- Protokolle: StopQuery
- Protokolle: PutSubscriptionFilter
- Protokolle: TestMetricFilter

Ein Benutzer, der die Konsole auch zum Verwalten von Protokoll-Abonnements verwendet, benötigt die folgenden Berechtigungen:

- ja: DescribeElasticsearchDomain
- ja: ListDomainNames
- ich bin: AttachRolePolicy
- ich bin: CreateRole
- ich bin: GetPolicy
- ich bin: GetPolicyVersion
- ich bin: GetRole
- ich bin: ListAttachedRolePolicies
- ich bin: ListRoles
- Kinese: DescribeStreams
- Kinese: ListStreams
- Lambda: AddPermission
- Lambda: CreateFunction
- Lambda: GetFunctionConfiguration
- Lambda: ListAliases
- Lambda: ListFunctions
- Lambda: ListVersionsByFunction
- Lambda: RemovePermission
- s3: ListBuckets

AWS verwaltete (vordefinierte) Richtlinien für CloudWatch Protokolle

AWS adressiert viele gängige Anwendungsfälle durch die Bereitstellung eigenständiger IAM-Richtlinien, die von erstellt und verwaltet werden. AWS Die verwalteten Richtlinien erteilen die erforderlichen Berechtigungen für viele häufige Anwendungsfälle, sodass Sie nicht mühsam ermitteln müssen, welche Berechtigungen erforderlich sind. Weitere Informationen finden Sie unter [AWS - verwaltete Richtlinien](#) im IAM-Benutzerhandbuch.

Die folgenden AWS verwalteten Richtlinien, die Sie Benutzern und Rollen in Ihrem Konto zuordnen können, gelten nur für CloudWatch Logs:

- CloudWatchLogsFullAccess— Gewährt vollen Zugriff auf CloudWatch Protokolle.

- **CloudWatchLogsReadOnlyAccess**— Gewährt schreibgeschützten Zugriff auf Logs. CloudWatch

CloudWatchLogsFullAccess

Die CloudWatchLogsFullAccessRichtlinie gewährt vollen Zugriff auf CloudWatch Protokolle. Die Richtlinie umfasst die `cloudwatch:GenerateQueryResultsSummary` Berechtigungen `cloudwatch:GenerateQuery` und, sodass Benutzer mit dieser Richtlinie anhand einer Aufforderung in natürlicher Sprache eine [CloudWatch Logs Insights-Abfragezeichenfolge](#) generieren können. Den vollständigen Inhalt der Richtlinie finden Sie [CloudWatchLogsFullAccess](#) im Referenzhandbuch für AWS verwaltete Richtlinien.

CloudWatchLogsReadOnlyAccess

Die CloudWatchLogsReadOnlyAccessRichtlinie gewährt nur Lesezugriff auf Protokolle. CloudWatch Sie umfasst die `cloudwatch:GenerateQueryResultsSummary` Berechtigungen `cloudwatch:GenerateQuery` und, sodass Benutzer mit dieser Richtlinie anhand einer Aufforderung in natürlicher Sprache eine [CloudWatch Logs Insights-Abfragezeichenfolge](#) generieren können. Den vollständigen Inhalt der Richtlinie finden Sie [CloudWatchLogsReadOnlyAccess](#) im Referenzhandbuch für AWS verwaltete Richtlinien.

CloudWatchOpenSearchDashboardsFullAccess

Die CloudWatchOpenSearchDashboardsFullAccessRichtlinie gewährt Zugriff auf das Erstellen, Verwalten und Löschen von Integrationen mit OpenSearch Service sowie auf die Erstellung von Löschprotokoll-Dashboards in diesen Integrationen. Weitere Informationen finden Sie unter [Analysieren Sie mit Amazon OpenSearch Service](#).

Den vollständigen Inhalt der Richtlinie finden Sie [CloudWatchOpenSearchDashboardsFullAccess](#) im Referenzhandbuch für AWS verwaltete Richtlinien.

CloudWatchOpenSearchDashboardAccess

Die CloudWatchOpenSearchDashboardAccessRichtlinie gewährt Zugriff auf Dashboards mit verkauften Protokollen, die mithilfe von Analysen erstellt wurden. Amazon OpenSearch Service Weitere Informationen finden Sie unter [Analysieren Sie mit Amazon OpenSearch Service](#).

Important

Damit eine Rolle oder ein Benutzer nicht nur diese Richtlinie gewähren kann, müssen Sie diese Dashboards auch angeben, wenn Sie die Integration mit Service erstellen. OpenSearch

Weitere Informationen finden Sie unter [Schritt 1: Erstellen Sie die Integration mit Service OpenSearch](#).

Den vollständigen Inhalt der Richtlinie finden Sie [CloudWatchOpenSearchDashboardAccess](#) im Referenzhandbuch für AWS verwaltete Richtlinien.

CloudWatchLogsCrossAccountSharingConfiguration

Die CloudWatchLogsCrossAccountSharingConfigurationRichtlinie gewährt Zugriff auf die Erstellung, Verwaltung und Anzeige von Observability Access Manager-Links zur gemeinsamen Nutzung von CloudWatch Logs-Ressourcen zwischen Konten. Weitere Informationen finden Sie unter [Kontoübergreifende Beobachtbarkeit von CloudWatch](#).

Den vollständigen Inhalt der Richtlinie finden Sie [CloudWatchLogsCrossAccountSharingConfiguration](#) im Referenzhandbuch für AWS verwaltete Richtlinien.

CloudWatchLogsAPIKeyAccess

Die CloudWatchLogsAPIKeyAccessRichtlinie aktiviert die CloudWatch Logs-API-Schlüsselauthentifizierung und die verschlüsselte Protokollaufnahme. Diese Richtlinie gewährt Berechtigungen zur Authentifizierung mithilfe von Bearer-Token und zum Schreiben von Protokollereignissen in Logs sowie zusätzliche AWS KMS Berechtigungen zum Entschlüsseln und Generieren von Datenschlüsseln, wenn CloudWatch Protokolle verschlüsselt werden.

Durch diese Richtlinie werden die folgenden Berechtigungen gewährt:

- **logs**— Ermöglicht Prinzipalen, sich über API-Schlüsselträgertoken zu authentifizieren und Protokollereignisse in Logs-Streams zu schreiben. CloudWatch
- **kms**— Ermöglicht Prinzipalen das Lesen von AWS KMS Schlüsselmetadaten, das Generieren von Datenschlüsseln für die Verschlüsselung und das Entschlüsseln von Daten. Diese Berechtigungen unterstützen verschlüsselte CloudWatch Protokolle, indem sie es dem Dienst ermöglichen, Protokolldaten mit vom Kunden verwalteten Schlüsseln zu verschlüsseln. AWS KMS Der Zugriff ist auf Vorgänge beschränkt, die über den CloudWatch Logs-Dienst aufgerufen werden.

Weitere Einzelheiten zu dieser Richtlinie, einschließlich der neuesten Version des JSON-Richtliniendokuments, finden Sie unter [CloudWatchLogsAPIKeyAccess](#) im AWS Referenzhandbuch für verwaltete Richtlinien.

CloudWatch Protokolliert Aktualisierungen für AWS Verwaltete Richtlinien

Zeigt Details zu Aktualisierungen der AWS verwalteten Richtlinien für CloudWatch Logs an, seit dieser Dienst begonnen hat, diese Änderungen zu verfolgen. Wenn Sie automatische Benachrichtigungen über Änderungen an dieser Seite erhalten möchten, abonnieren Sie den RSS-Feed auf der Seite mit dem Verlauf der CloudWatch Protokolldokumente.

Änderungen	Beschreibung	Date
CloudWatchLogsAPIKeyAccess - Neue Richtlinie.	CloudWatch Logs hat eine neue verwaltete Richtlinie hinzugefügt CloudWatchLogsAPIKeyAccess. Diese Richtlinie ermöglicht die CloudWatch Logs-API-Schlüsselauthentifizierung und die verschlüsselte Protokollaufnahme und gewährt Berechtigungen zur Authentifizierung mithilfe von Bearer-Token und zum Schreiben von Protokollereignissen in Logs. CloudWatch	17. Februar 2026
CloudWatchLogsFullAccess – Aktualisierung auf eine bestehende Richtlinie.	CloudWatch Logs, denen Berechtigungen hinzugefügt CloudWatchLogsFullAccess wurden. Berechtigungen für Aktionen zur Observability-Verwaltung wurden hinzugefügt, um einen schreibgeschützten Zugriff auf Telemetrie-Pipelines und	02. Dezember 2025

Änderungen	Beschreibung	Date
	S3-Tabellenintegrationen zu ermöglichen.	
CloudWatchLogsReadOnlyAccess – Aktualisierung auf eine bestehende Richtlinie.	CloudWatch Logs, denen Berechtigungen hinzugefügt wurden CloudWatchLogsRead OnlyAccess. Berechtigungen für Aktionen zur Observability-Verwaltung wurden hinzugefügt, um einen schreibgeschützten Zugriff auf Telemetrie-Pipelines und S3-Tabellenintegrationen zu ermöglichen.	02. Dezember 2025
CloudWatchLogsFullAccess – Aktualisierung auf eine bestehende Richtlinie.	CloudWatch Logs, denen Berechtigungen hinzugefügt wurden CloudWatchLogsFull Access. Berechtigungen für <code>cloudwatch:GenerateQueryResultsSummary</code> wurden hinzugefügt, um die Generierung einer Zusammenfassung der Abfrageergebnisse in natürlicher Sprache zu ermöglichen.	20. Mai 2025

Änderungen	Beschreibung	Date
CloudWatchLogsReadOnlyAccess – Aktualisierung auf eine bestehende Richtlinie.	CloudWatch Protokolliert hinzugefügte Berechtigungen für CloudWatchLogsRead OnlyAccess. Berechtigungen für <code>cloudwatch:GenerateQueryResultsSummary</code> wurden hinzugefügt, um die Generierung einer Zusammenfassung der Abfrageergebnisse in natürlicher Sprache zu ermöglichen.	20. Mai 2025
CloudWatchLogsFullAccess – Aktualisierung auf eine bestehende Richtlinie.	CloudWatch Protokolliert hinzugefügte Berechtigungen für CloudWatchLogsFull Access. Es wurden Berechtigungen für Amazon OpenSearch Service und IAM hinzugefügt, um die Integration von CloudWatch Logs in den OpenSearch Service für einige Funktionen zu ermöglichen.	01. Dezember 2024

Änderungen	Beschreibung	Date
CloudWatchOpenSearchDashboardsFullAccess — Neue IAM-Richtlinie.	CloudWatch Logs hat eine neue IAM-Richtlinie hinzugefügt, CloudWatchOpenSearchDashboardsFullAccess.- Diese Richtlinie gewährt Zugriff auf das Erstellen, Verwalten und Löschen von Integrationen mit OpenSearch Service sowie auf das Erstellen, Verwalten und Löschen von Dashboards für verkaufte Protokolle in diesen Integrationen. Weitere Informationen finden Sie unter Analysieren Sie mit Amazon OpenSearch Service .	01. Dezember 2024
CloudWatchOpenSearchDashboardAccess — Neue IAM-Richtlinie.	CloudWatch Logs hat eine neue IAM-Richtlinie hinzugefügt, CloudWatchOpenSearchDashboardAccess.- Diese Richtlinie gewährt Zugriff auf Dashboards für verkaufte Logs, die von bereitgestellt werden. Amazon OpenSearch Service Weitere Informationen finden Sie unter Analysieren Sie mit Amazon OpenSearch Service .	01. Dezember 2024

Änderungen	Beschreibung	Date
CloudWatchLogsFullAccess – Aktualisierung auf eine bestehende Richtlinie.	CloudWatch Für Logs wurde eine Berechtigung hinzugefügt. CloudWatchLogsFullAccess Die <code>cloudwatch:GenerateQuery</code> Berechtigung wurde hinzugefügt, sodass Benutzer mit dieser Richtlinie anhand einer Aufforderung in natürlicher Sprache eine CloudWatch Logs Insights-Abfragezeilenfolge generieren können.	27. November 2023
CloudWatchLogsReadOnlyAccess – Aktualisierung auf eine bestehende Richtlinie.	CloudWatch hat eine Berechtigung zu hinzugefügt CloudWatchLogsReadOnlyAccess. Die <code>cloudwatch:GenerateQuery</code> Berechtigung wurde hinzugefügt, sodass Benutzer mit dieser Richtlinie anhand einer Aufforderung in natürlicher Sprache eine CloudWatch Logs Insights-Abfragezeilenfolge generieren können.	27. November 2023

Änderungen	Beschreibung	Date
CloudWatchLogsReadOnlyAccess – Aktualisierung auf eine bestehende Richtlinie	CloudWatch Logs haben Berechtigungen für hinzugefügt CloudWatchLogsReadOnlyAccess. Die logs:StopLiveTail Berechtigungen logs:StartLiveTail und wurden hinzugefügt, sodass Benutzer mit dieser Richtlinie die Konsole verwenden können, um CloudWatch Logs-Live-Tail-Sitzungen zu starten und zu beenden. Weitere Informationen finden Sie unter Use live tail to view logs in near real time.	6. Juni 2023
CloudWatchLogsCrossAccountSharingConfiguration – Neue Richtlinie	CloudWatch Logs hat eine neue Richtlinie hinzugefügt, mit der Sie CloudWatch kontoübergreifende Observability-Links verwalten können, die CloudWatch Log-Protokollgruppen gemeinsam nutzen. Weitere Informationen finden Sie unter kontoübergreifende Beobachtbarkeit CloudWatch	27. November 2022

Änderungen	Beschreibung	Date
CloudWatchLogsReadOnlyAccess – Aktualisierung auf eine bestehende Richtlinie	CloudWatch Protokolliert hinzugefügte Berechtigungen für. CloudWatchLogsReadOnlyAccess Die <code>oam:ListAttachedLinks</code> Berechtigungen <code>oam:ListSinks</code> und wurden hinzugefügt, sodass Benutzer mit dieser Richtlinie die Konsole verwenden können, um Daten, die von Quellkonten gemeinsam genutzt wurden, CloudWatch kontenübergreifend anzusehen.	27. November 2022

Beispiele für vom Kunden verwaltete Richtlinien

Sie können Ihre eigenen benutzerdefinierten IAM-Richtlinien erstellen, um Berechtigungen für CloudWatch Logs-Aktionen und -Ressourcen zu gewähren. Die benutzerdefinierten Richtlinien können Sie dann den -Benutzern oder -Gruppen zuweisen, die diese Berechtigungen benötigen.

In diesem Abschnitt finden Sie Beispielbenutzerrichtlinien, die Berechtigungen für verschiedene CloudWatch Logs-Aktionen gewähren. Diese Richtlinien funktionieren, wenn Sie die CloudWatch Logs-API, AWS SDKs oder die AWS CLI verwenden.

Beispiele

- [Beispiel 1: Vollzugriff auf Logs zulassen CloudWatch](#)
- [Beispiel 2: Erlauben Sie den schreibgeschützten Zugriff auf Protokolle CloudWatch](#)
- [Beispiel 3: Erlaubt den Zugriff auf eine Protokollgruppe//Protokollstream](#)

Beispiel 1: Vollzugriff auf Logs zulassen CloudWatch

Die folgende Richtlinie ermöglicht einem Benutzer den Zugriff auf alle CloudWatch Logs-Aktionen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Beispiel 2: Erlauben Sie den schreibgeschützten Zugriff auf Protokolle CloudWatch

AWS stellt eine CloudWatchLogsReadOnlyAccessRichtlinie bereit, die den schreibgeschützten Zugriff auf Protokolldaten ermöglicht. CloudWatch Diese Richtlinie umfasst die folgenden Berechtigungen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:Describe*",
        "logs:Get*",
        "logs:List*",
        "logs:StartQuery",
        "logs:StopQuery",
        "logs:TestMetricFilter",
        "logs:FilterLogEvents",
        "logs:StartLiveTail",
        "logs:StopLiveTail",
        "cloudwatch:GenerateQuery"
      ]
    }
  ]
}
```

```

    ],
    "Effect": "Allow",
    "Resource": "*"
  }
]
}

```

Beispiel 3: Erlaubt den Zugriff auf eine Protokollgruppe//Protokollstream

CloudWatch Logs hat zwei Ressourcentypen mit unterschiedlichen ARN-Formaten:

- Protokollgruppe: `arn:aws:logs:region:account:log-group:LogGroupName`
- Datenstrom protokollieren: `arn:aws:logs:region:account:log-group:LogGroupName:log-stream:StreamName`

Beim Schreiben von IAM-Richtlinien muss das von Ihnen verwendete ARN-Format dem Ressourcentyp entsprechen, für den die API autorisiert. In der [Referenz zur Serviceautorisierung](#) finden Sie den Ressourcentyp, den jede API benötigt.

Beispiel 3a: Erlauben Sie den Zugriff auf Aktionen auf Protokollgruppenebene für eine bestimmte Protokollgruppe

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:DeleteLogGroup",
        "logs:PutRetentionPolicy",
        "logs:PutSubscriptionFilter",
        "logs:DescribeLogStreams"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:us-west-2:123456789012:log-group:SampleLogGroupName"
    }
  ]
}

```

Diese Aktionen autorisieren für den Ressourcentyp. `log-group` Das Standard-ARN-Format (ohne `:` `*`) wird unterstützt.

Beispiel 3b: Erlauben Sie den Zugriff auf Aktionen auf Logstream-Ebene für eine bestimmte Protokollgruppe

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:us-west-2:123456789012:log-group:SampleLogGroupName:*"
    }
  ]
}
```

Diese Aktionen autorisieren für den Ressourcentyp. `log-stream` Das `:*` Suffix ist erforderlich, um allen Protokollströmen innerhalb der Protokollgruppe zuzuordnen oder einen bestimmten Stream mit anzugeben. `:log-stream:` *StreamName*

Beispiel 3c: Kombinierte Richtlinie für Protokollgruppen- und Logstream-Aktionen

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs>DeleteLogGroup",
        "logs:PutRetentionPolicy",
        "logs:DescribeLogStreams",
        "logs:FilterLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:us-west-2:123456789012:log-group:SampleLogGroupName"
    },
    {
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:us-west-2:123456789012:log-stream:SampleLogGroupName:SampleLogStreamName"
    }
  ]
}
```

```
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:logs:us-west-2:123456789012:log-
group:SampleLogGroupName:*"
  }
]
```

Markierung und IAM-Richtlinien für die Steuerung auf der Protokollgruppenebene verwenden

Sie können Benutzern Zugriff auf bestimmte Protokollgruppen gewähren und sie gleichzeitig daran hindern, auf andere Protokollgruppen zuzugreifen. Hierzu markieren Sie Ihre Protokollgruppen und verwenden IAM-Richtlinien, die auf diese Tags verweisen. Um Tags auf eine Protokollgruppe anzuwenden, benötigen Sie entweder die `logs:TagResource-` oder `logs:TagLogGroup-` Berechtigung. Dies gilt sowohl, wenn Sie der Protokollgruppe bei der Erstellung Tags zuweisen, als auch, wenn Sie die Tags später zuweisen.

Weitere Informationen zum Taggen von Protokollgruppen finden Sie unter [Taggen Sie Protokollgruppen in Amazon CloudWatch Logs](#).

Wenn Sie Protokollgruppen markieren, können Sie einem Benutzer eine IAM-Richtlinie erteilen, um nur Zugriff auf die Protokollgruppen mit einem bestimmten Tag zu gewähren. Beispiel: Die folgende Richtlinienanweisung gewährt nur den Zugriff auf Regeln mit dem Wert `Green` für den Tag-Schlüssel `Team`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:*"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/Team": "Green"
        }
      }
    }
  ]
}
```

```

    }
  ]
}
```

Die Operationen `StopQuery` und die `StopLiveTailAPI` interagieren nicht mit AWS Ressourcen im herkömmlichen Sinne. Sie geben keine Daten zurück, legen keine Daten ab und ändern keine Ressource in irgendeiner Weise. Stattdessen funktionieren sie nur bei einer bestimmten Live-Tail-Sitzung oder einer bestimmten CloudWatch Logs Insights-Abfrage, die nicht als Ressourcen kategorisiert sind. Wenn Sie das Feld `Resource` in den IAM-Richtlinien für diese Operationen angeben, müssen Sie daher den Wert des Felds `Resource` wie im folgenden Beispiel auf `*` festlegen.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:StopQuery",
        "logs:StopLiveTail"
      ],
      "Resource": "*"
    }
  ]
}
```

Weitere Informationen zur Verwendung von IAM-Richtlinienanweisungen finden Sie unter [Steuern des Zugriffs mithilfe von Richtlinien](#) im IAM-Benutzerhandbuch.

CloudWatch Referenz zu Protokollberechtigungen

Wenn Sie die [Zugriffskontrolle](#) einrichten und Berechtigungsrichtlinien für eine IAM-Identität (identitätsbasierte Richtlinie) verfassen, können Sie die folgende Tabelle als Referenz verwenden. In der Tabelle sind die einzelnen CloudWatch Logs-API-Operationen und die entsprechenden Aktionen aufgeführt, für die Sie Berechtigungen zur Ausführung der Aktion erteilen können. Sie

geben die Aktionen im Feld **Action** der Richtlinie an. Für das **Resource** Feld können Sie den ARN einer Protokollgruppe oder eines Protokolldatenstroms angeben oder angeben, * dass er für alle CloudWatch Logs-Ressourcen steht.

Sie können in Ihren AWS CloudWatch Logs-Richtlinien Bedingungsschlüssel für alle Bereiche verwenden, um Bedingungen auszudrücken. Eine vollständige Liste der AWS Schlüssel für alle Bedingungen finden Sie unter [AWS Globale Schlüssel und IAM-Bedingungskontextschlüssel](#) im IAM-Benutzerhandbuch.

 Note

Um eine Aktion anzugeben, verwenden Sie das Präfix `logs:` gefolgt vom Namen der API-Operation. Zum Beispiel: `logs:CreateLogGroup`, `logs:CreateLogStream`, oder `logs:*` (für alle CloudWatch Logs-Aktionen).

CloudWatch Protokolliert API-Operationen und die erforderlichen Berechtigungen für Aktionen

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
CancelExportTask	<code>logs:CancelExportTask</code> Erforderlich zum Abbrechen einer ausstehenden oder laufenden Exportaufgabe.
CreateExportTask	<code>logs:CreateExportTask</code> Erforderlich zum Exportieren von Daten aus einer Protokollgruppe in einen Amazon-S3-Bucket.
CreateLogGroup	<code>logs:CreateLogGroup</code> Erforderlich zum Erstellen einer neuen Protokollgruppe.
CreateLogStream	<code>logs:CreateLogStream</code>

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
	Erforderlich zum Erstellen eines neuen Protokoll-Stream in eine Protokollgruppe.
DeleteDestination	<code>logs:DeleteDestination</code> Erforderlich zum Löschen eines Protokollziels und Aktivieren von Abonnementfiltern für das Ziel.
DeleteLogGroup	<code>logs:DeleteLogGroup</code> Erforderlich zum Löschen einer Protokollgruppe und der jeweiligen archivierten Protokollereignisse.
DeleteLogStream	<code>logs:DeleteLogStream</code> Erforderlich zum Löschen eines Protokoll-Stream und der jeweiligen archivierten Protokollereignisse.
DeleteMetricFilter	<code>logs:DeleteMetricFilter</code> Erforderlich zum Löschen eines Metrikfilters für eine Protokollgruppe.
DeleteQueryDefinition	<code>logs:DeleteQueryDefinition</code> Erforderlich, um eine gespeicherte Abfragedefinition in CloudWatch Logs Insights zu löschen.

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
DeleteResourcePolicy	<code>logs:DeleteResourcePolicy</code> Erforderlich, um eine CloudWatch Logs-Ressourcenrichtlinie zu löschen.
DeleteRetentionPolicy	<code>logs:DeleteRetentionPolicy</code> Erforderlich zum Löschen der Aufbewahrungsrichtlinie einer Protokollgruppe.
DeleteSubscriptionFilter	<code>logs:DeleteSubscriptionFilter</code> Erforderlich zum Löschen des Abonnementfilters für eine Protokollgruppe.
DescribeDestinations	<code>logs:DescribeDestinations</code> Erforderlich zum Anzeigen aller Ziele für ein Konto.
DescribeExportTasks	<code>logs:DescribeExportTasks</code> Erforderlich zum Anzeigen aller Exportaufgaben für das Konto.
DescribeLogGroups	<code>logs:DescribeLogGroups</code> Erforderlich zum Anzeigen aller Protokollgruppen für ein Konto.
DescribeLogStreams	<code>logs:DescribeLogStreams</code> Erforderlich zum Anzeigen aller Protokollstreams für eine Protokollgruppe.

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
DescribeMetricFilters	<code>logs:DescribeMetricFilters</code> Erforderlich zum Anzeigen aller Metriken für eine Protokollgruppe.
DescribeQueryDefinitions	<code>logs:DescribeQueryDefinitions</code> Erforderlich, um die Liste der gespeicherten Abfragedefinitionen in CloudWatch Logs Insights zu sehen.
DescribeQueries	<code>logs:DescribeQueries</code> Erforderlich, um die Liste der CloudWatch Logs Insights-Abfragen zu sehen, die geplant sind, ausgeführt werden oder kürzlich ausgeführt wurden.
DescribeResourcePolicies	<code>logs:DescribeResourcePolicies</code> Erforderlich, um eine Liste der CloudWatch Logs-Ressourcenrichtlinien anzuzeigen.
DescribeSubscriptionFilters	<code>logs:DescribeSubscriptionFilters</code> Erforderlich zum Anzeigen aller Abonnemen tfilter für eine Protokollgruppe.
FilterLogEvents	<code>logs:FilterLogEvents</code> Erforderlich zum Sortieren von Protokoll ereignissen nach Gruppenfiltermuster.

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
GetLogEvents	<code>logs: GetLogEvents</code> Erforderlich zum Abrufen von Protokollereignissen aus einem Protokoll-Stream.
GetLogGroupFields	<code>logs: GetLogGroupFields</code> Erforderlich, um die Liste der Felder abzurufen, die in den Protokollereignissen in einer Protokollgruppe enthalten sind.
GetLogRecord	<code>logs: GetLogRecord</code> Erforderlich, um die Details aus einem einzelnen Protokollereignis abzurufen.
GetLogObject	<code>logs: GetLogRecord</code> Erforderlich, um den Inhalt großer Teile von Protokollereignissen abzurufen, die über die PutOpenTelemetryLogs API aufgenommen wurden.
GetQueryResults	<code>logs: GetQueryResults</code> Erforderlich, um die Ergebnisse von CloudWatch Logs Insights-Abfragen abzurufen.
<code>ListEntitiesForLogGroup</code> (Berechtigung CloudWatch nur für Konsolen)	<code>logs: ListEntitiesForLogGroup</code> Erforderlich, um die Entitäten zu finden, die einer Protokollgruppe zugeordnet sind. Erforderlich, um verwandte Logs in der CloudWatch Konsole zu durchsuchen.

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
ListLogGroupsForEntity (Erlaubnis CloudWatch nur für die Konsole)	logs:ListLogGroupsForEntity Erforderlich, um die mit einer Entität verknüpften Protokollgruppen zu finden. Erforderlich, um verwandte Logs in der CloudWatch Konsole zu durchsuchen.
ListTagsLogGroup	logs:ListTagsLogGroup Erforderlich zum Auflisten der mit einer Protokollgruppe verbundenen Tags.
ListLogGroups	logs:DescribeLogGroups Erforderlich zum Anzeigen aller Protokollgruppen für ein Konto.
ProcessWithPipeline (Nur mit Genehmigung)	logs:ProcessWithPipeline Erforderlich, um Protokollereignisse vor der Speicherung durch Pipeline-Transformatoren zu verarbeiten und umzuwandeln. Diese Berechtigung muss für die IAM-Rolle erteilt werden, die in einer Pipeline-Konfiguration als CloudWatch Protokollquellenrolle übergeben wurde. Die Ressource ist die Protokollgruppe ARN.
PutDestination	logs:PutDestination Erforderlich zum Erstellen oder Aktualisieren eines Ziel-Protokoll-Streams (z. B. ein Kinesis-Stream).

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
PutDestinationPolicy	<code>logs:PutDestinationPolicy</code> Erforderlich zum Erstellen oder Aktualisieren einer Zugriffsrichtlinie im Zusammenhang mit einem vorhandenen Protokollziel.
PutLogEvents	<code>logs:PutLogEvents</code> Erforderlich für den Upload eines Batches von Protokollereignissen in einen Protokoll-Stream.
PutMetricFilter	<code>logs:PutMetricFilter</code> Erforderlich zum Erstellen oder Aktualisieren eines Metrikfilters, der einer Protokollgruppe zugeordnet wird.
PutQueryDefinition	<code>logs:PutQueryDefinition</code> Erforderlich, um eine Abfrage in CloudWatch Logs Insights zu speichern, einschließlich gespeicherter Abfragen mit Parametern.
PutResourcePolicy	<code>logs:PutResourcePolicy</code> Erforderlich, um eine CloudWatch Logs-Ressourcenrichtlinie zu erstellen.
PutRetentionPolicy	<code>logs:PutRetentionPolicy</code> Erforderlich zum Festlegen der Anzahl der Tage, die Protokollereignisse (Aufbewahrung) in einer Protokollgruppe aufbewahrt werden sollen.

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
PutSubscriptionFilter	<code>logs:PutSubscriptionFilter</code> Erforderlich zum Erstellen oder Aktualisieren eines Abonnementfilters, der einer Protokollgruppe zugeordnet wird.
StartQuery	<code>logs:StartQuery</code> Erforderlich, um CloudWatch Logs Insights-Abfragen zu starten. Um eine gespeicherte Abfrage mit Parametern auszuführen, benötigen Sie außerdem <code>logs:DescribeQueryDefinitions</code>.
StopQuery	<code>logs:StopQuery</code> Erforderlich, um eine laufende CloudWatch Logs Insights-Abfrage zu beenden.
TagLogGroup	<code>logs:TagLogGroup</code> Erforderlich zum Hinzufügen oder Aktualisieren von Protokollgruppen-Tags.
TestMetricFilter	<code>logs:TestMetricFilter</code> Erforderlich zum Testen eines Filtermusters anhand einer Stichprobe von Protokollereignis-Nachrichten.

Verwenden von serviceverknüpften Rollen für Logs CloudWatch

Amazon CloudWatch Logs verwendet AWS Identity and Access Management (IAM) [serviceverknüpfte Rollen](#). Eine serviceverknüpfte Rolle ist eine einzigartige Art von IAM-Rolle, die

direkt mit Logs verknüpft ist. CloudWatch Service-linked Rollen sind in CloudWatch Logs vordefiniert und enthalten alle Berechtigungen, die der Dienst benötigt, um andere AWS Dienste in Ihrem Namen aufzurufen.

Eine dienstbezogene Rolle macht die Einrichtung von CloudWatch Logs effizienter, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. CloudWatch Logs definiert die Berechtigungen seiner dienstbezogenen Rollen, und sofern nicht anders definiert, können nur CloudWatch Logs diese Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann an keine andere IAM-Entität angefügt werden.

Weitere Informationen zu anderen Services, die serviceverknüpfte Rollen unterstützen, finden Sie unter [AWS -Services, die mit IAM funktionieren](#). Suchen Sie in der Spalte Service-LinkedRolle nach den Diensten, für die Ja angegeben ist. Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Service-linked Rollenberechtigungen für CloudWatch Logs

CloudWatch Logs verwendet die mit dem Dienst verknüpfte Rolle namens `AWSServiceRoleForLogDelivery`. CloudWatch Logs verwendet diese dienstbezogene Rolle, um Protokolle direkt in Firehose zu schreiben. Weitere Informationen finden Sie unter [Protokollierung aktivieren von AWS service](#).

Die serviceverknüpfte Rolle `AWSServiceRoleForLogDelivery` vertraut darauf, dass die folgenden Services die Rolle annehmen:

- `logs.amazonaws.com`

Die Richtlinie für Rollenberechtigungen ermöglicht es CloudWatch Logs `AWSServiceRoleForLogDeliveryPolicy`, die folgenden Aktionen an den angegebenen Ressourcen durchzuführen. Das vollständige JSON-Richtliniendokument finden Sie [AWSServiceRoleForLogDeliveryPolicy](#) im Referenzhandbuch für AWS verwaltete Richtlinien.

- Aktion: `firehose:PutRecord``firehose:PutRecordBatch`, und `firehose:ListTagsForDeliveryStream` für alle Firehose-Lieferdatenströme, die ein Tag mit einem `LogDeliveryEnabled` Schlüssel mit einem Wert von `true` haben. Dieses Tag wird automatisch an einen Firehose-Lieferstream angehängt, wenn Sie ein Abonnement für die Übermittlung der Protokolle an Firehose erstellen.

- Aktion: `kms:GenerateDataKey` und `kms:Decrypt` für alle AWS KMS Schlüssel, aber nur, wenn die Anfrage über Firehose gestellt wird (erzwungen durch den auf gesetzten `kms:ViaService` Bedingungsschlüssel). `firehose.*.amazonaws.com` Diese Berechtigungen ermöglichen es CloudWatch Logs, Protokolle an Firehose-Lieferdatenströme zu senden, die serverseitige Verschlüsselung mit vom Kunden verwalteten Schlüsseln () SSE-CMK verwenden. Die AWS KMS Schlüsselrichtlinie des Kunden muss außerdem eigenständig Zugriff auf die mit dem Service verknüpfte Rolle gewähren.

Sie müssen Berechtigungen konfigurieren, die es einer IAM-Entität erlaubt, eine serviceverknüpfte Rolle zu erstellen, zu bearbeiten oder zu löschen. Diese Entität kann ein Benutzer, eine Gruppe oder eine Rolle sein. Weitere Informationen finden Sie unter [Service-Linked Rollenberechtigungen](#) im IAM-Benutzerhandbuch.

Eine serviceverknüpfte Rolle für Logs erstellen CloudWatch

Sie brauchen keine serviceverknüpfte Rolle manuell erstellen. Wenn Sie Protokolle so einrichten, dass sie direkt an einen Firehose-Stream in der AWS-Managementkonsole, der oder der AWS CLI AWS API gesendet werden, erstellt CloudWatch Logs die dienstbezogene Rolle für Sie.

Wenn Sie diese serviceverknüpfte Rolle löschen und sie dann erneut erstellen müssen, können Sie dasselbe Verfahren anwenden, um die Rolle in Ihrem Konto neu anzulegen. Wenn Sie erneut Protokolle so einrichten, dass sie direkt an einen Firehose-Stream gesendet werden, erstellt CloudWatch Logs die dienstverknüpfte Rolle erneut für Sie.

Bearbeitung einer serviceverknüpften Rolle für Logs CloudWatch

CloudWatch Logs ermöglicht es Ihnen nicht `AWSServiceRoleForLogDelivery`, diese oder jede andere dienstbezogene Rolle zu bearbeiten, nachdem Sie sie erstellt haben. Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollename nach der Erstellung nicht geändert werden. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter [Service-Linked Rolle bearbeiten](#).

Löschen einer serviceverknüpften Rolle für Logs CloudWatch

Wenn Sie ein Feature oder einen Dienst, die bzw. der eine serviceverknüpften Rolle erfordert, nicht mehr benötigen, sollten Sie diese Rolle löschen. Auf diese Weise haben Sie keine ungenutzte juristische Stelle, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch die Ressourcen für Ihre serviceverknüpften Rolle zunächst bereinigen, bevor Sie sie manuell löschen können.

Note

Wenn der CloudWatch Logs-Dienst die Rolle verwendet, wenn Sie versuchen, die Ressourcen zu löschen, schlägt das Löschen möglicherweise fehl. Wenn dies passiert, warten Sie einige Minuten und versuchen Sie es erneut.

Um CloudWatch Logs-Ressourcen zu löschen, die verwendet werden von `AWSServiceRoleForLogDelivery` Serviceverknüpfte Rolle

- Beenden Sie das direkte Senden von Protokollen an Firehose-Streams.

So löschen Sie die serviceverknüpfte Rolle mit IAM

Verwenden Sie die IAM-Konsole, die oder die AWS API AWS CLI, um die `AWSServiceRoleForLogDelivery` serviceverknüpfte Rolle zu löschen. Weitere Informationen finden Sie unter Rolle [löschen Service-Linked](#)

Unterstützte Regionen für Rollen, die mit dem CloudWatch Logs-Dienst verknüpft sind

CloudWatch Logs unterstützt die Verwendung von dienstbezogenen Rollen in allen AWS Regionen, in denen der Dienst verfügbar ist. Weitere Informationen finden Sie unter [CloudWatch Logs, Regionen und Endpoints](#).

CloudWatch Protokolliert Aktualisierungen für AWS Mit dem Dienst verknüpfte Rollen

Hier finden Sie Informationen zu Aktualisierungen der AWS dienstverknüpften Rolle für CloudWatch Logs, seit dieser Dienst mit der Erfassung dieser Änderungen begonnen hat. Wenn Sie automatische Benachrichtigungen über Änderungen an dieser Seite erhalten möchten, abonnieren Sie den RSS-Feed auf der Seite mit dem Verlauf der CloudWatch Protokolldokumente.

Änderungen	Beschreibung	Date
AWSServiceRoleForLogDelivery Richtlinie für dienstbezogene Rollen	CloudWatch Hinzugefügte Protokolle <code>kms:GenerateDataKey</code> und <code>kms:Decrypt</code> Berechtig	15. Mai 2026

Änderungen	Beschreibung	Date
— Aktualisierung einer bestehenden Richtlinie	ungen für die IAM-Richtlinie, die mit der <code>AWSServiceRoleForLogDelivery</code> service verknüpften Rolle verknüpft sind. Diese Berechtigungen sind mit dem <code>kms:ViaService</code> Bedingungsschlüssel so begrenzt, dass sie nur über Firehose verwendet werden können. Diese Änderung ermöglicht CloudWatch Logs die Übermittlung von Protokollen an Firehose-Lieferdatenströme, die serverseitige Verschlüsselung mit vom Kunden verwalteten Schlüsseln () SSE-CMK verwenden.	

Änderungen	Beschreibung	Date
AWSServiceRoleForLogDelivery Richtlinie für dienstbezogene Rollen — Aktualisierung einer bestehenden Richtlinie	CloudWatch Logs haben die Berechtigungen in der IAM-Richtlinie geändert, die der AWSServiceRoleForLogDeliverydienstbezogenen Rolle zugeordnet sind. Die folgende Änderung wurde vorgenommen: • Der firehose: ResourceTag/LogDeliveryEnabled": "true"-Bedingungsschlüssel wurde in aws:ResourceTag/LogDeliveryEnabled": "true" geändert.	15. Juli 2021
CloudWatch Logs haben begonnen, Änderungen nachzuverfolgen	CloudWatch Logs begann, Änderungen für die von AWS ihm verwalteten Richtlinien nachzuverfolgen.	10. Juni 2021

Konformitätsprüfung für Amazon CloudWatch Logs

Informationen darüber, ob AWS-Service ein [AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter](#) . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter [AWS Compliance-Programme AWS](#) .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter [Berichte herunterladen unter](#).

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der [AWS Sicherheitsdokumentation](#).

Resilienz in Amazon CloudWatch Logs

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. Regionen stellen mehrere physisch getrennte und isolierte Availability Zones bereit, die über hoch redundante Netzwerke mit niedriger Latenz und hohen Durchsätzen verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter [AWS Globale Infrastruktur](#).

Infrastruktursicherheit in Amazon CloudWatch Logs

Als verwalteter Service ist Amazon CloudWatch Logs durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter [AWS Cloud-Sicherheit](#). Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter [Infrastructure Protection](#) in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf CloudWatch Protokolle zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Verwenden von CloudWatch Protokollen mit VPC-Endpunkten der Schnittstelle

Wenn Sie Amazon Virtual Private Cloud (Amazon VPC) zum Hosten Ihrer AWS Ressourcen verwenden, können Sie eine private Verbindung zwischen Ihrer VPC und CloudWatch Logs herstellen. Sie können diese Verbindung verwenden, um Protokolle an Logs zu CloudWatch senden, ohne sie über das Internet zu senden. CloudWatch Logs unterstützt IPv4 VPC-Endpunkte in allen Regionen und unterstützt IPv6 Endpunkte in allen Regionen.

Amazon VPC ist ein AWS Service, mit dem Sie AWS Ressourcen in einem von Ihnen definierten virtuellen Netzwerk starten können. Mit einer VPC haben Sie die Kontrolle über Ihre Netzwerkeinstellungen, wie IP-Adressbereich, Subnetze, Routing-Tabellen und Netzwerk-Gateways. Um Ihre VPC mit CloudWatch Logs zu verbinden, definieren Sie einen VPC-Schnittstellen-Endpunkt für CloudWatch Logs. Mit dieser Art Endpunkt können Sie Ihre VPC mit AWS -Services verbinden. Der Endpunkt bietet zuverlässige, skalierbare Konnektivität zu CloudWatch Logs, ohne dass ein Internet-Gateway, eine NAT-Instanz (Network Address Translation) oder eine VPN-Verbindung erforderlich sind. Weitere Informationen finden Sie unter [Was ist Amazon VPC](#) im Benutzerhandbuch zu Amazon VPC.

Schnittstelle, auf der VPC-Endpunkte basieren AWS PrivateLink, eine AWS Technologie, die private Kommunikation zwischen AWS Diensten über eine elastic network interface mit privaten IP-Adressen ermöglicht. Weitere Informationen finden Sie unter [Neu — AWS PrivateLink für AWS Dienste](#).

Die folgenden Schritte sind für Benutzer von Amazon VPC vorgesehen. Weitere Informationen finden Sie unter [Erste Schritte](#) im Amazon VPC Benutzerhandbuch.

Verfügbarkeit

CloudWatch Logs unterstützt derzeit VPC-Endpunkte in allen AWS Regionen, einschließlich der AWS GovCloud (US) Regionen.

Einen VPC-Endpunkt für CloudWatch Logs erstellen

Um CloudWatch Logs mit Ihrer VPC zu verwenden, erstellen Sie einen VPC-Schnittstellen-Endpunkt für CloudWatch Logs. Der Service, den Sie wählen können, ist `com.amazonaws.Region.logs`. Um eine Verbindung mit einem FIPS-Endpunkt herzustellen, steht der Dienst zur Auswahl. `com.amazonaws.Region.logs-fips` Sie müssen keine Einstellungen für CloudWatch Logs

ändern. Weitere Informationen finden Sie unter [Erstellen eines Schnittstellenendpunkts](#) im Amazon VPC Leitfaden.

Einige CloudWatch Protokolle APIs, wie StartLiveTail und GetLogObject, werden unter einem anderen Endpunkt und VPC-Endpunkt gehostet: `stream-logs.Region.amazonaws.com`. Um einen VPC-Schnittstellen-Endpunkt für diese zu erstellen APIs, müssen Sie den Dienst `com.amazonaws` auswählen. **Region**.`stream-logs`. Um eine Verbindung mit einem FIPS-Endpunkt herzustellen, steht der Dienst zur Auswahl. `com.amazonaws.Region.stream-logs-fips`

Testen der Verbindung zwischen Ihrer VPC und Logs CloudWatch

Nachdem Sie den Endpunkt erstellt haben, können Sie die Verbindung testen.

Um die Verbindung zwischen Ihrer VPC und Ihrem CloudWatch Logs-Endpunkt zu testen

1. Stellen Sie eine Verbindung mit einer Amazon EC2-Instance in Ihrer VPC her. Weitere Informationen zum Herstellen einer Verbindung finden Sie unter [Herstellen einer Verbindung mit Ihrer Linux-Instance](#) or [Herstellen einer Verbindung mit Ihrer Windows-Instance](#) in der Amazon EC2-Dokumentation.
2. Verwenden Sie in der Instance den, AWS CLI um einen Protokolleintrag in einer Ihrer vorhandenen Protokollgruppen zu erstellen.

Zuerst erstellen Sie eine JSON-Datei mit einem Protokollereignis. Der Zeitstempel muss als Millisekunden seit dem 1. Januar 1970 00:00:00 UTC angegeben werden.

```
[
  {
    "timestamp": 1533854071310,
    "message": "VPC Connection Test"
  }
]
```

Verwenden Sie anschließend den `put-log-events`-Befehl zum Erstellen des Protokolleintrags:

```
aws logs put-log-events --log-group-name LogGroupName --log-stream-  
name LogStreamName --log-events file://JSONFileName
```


Wenn die Antwort auf den Befehl `nextSequenceToken` enthält, war der Befehl erfolgreich und Ihr VPC-Endpunkt funktioniert.

Steuern des Zugriffs auf Ihren CloudWatch Logs-VPC-Endpunkt

Eine VPC-Endpunktrichtlinie ist eine IAM-Ressourcenrichtlinie, die Sie einem Endpunkt beim Erstellen oder Ändern des Endpunkts zuordnen. Wenn Sie einem Endpunkt beim Erstellen keine Richtlinie zuordnen, wird ihm eine Standardrichtlinie mit Vollzugriff auf den Service zugeordnet. IAM-Richtlinien oder servicespezifische Richtlinien werden von einer Endpunktrichtlinie nicht überschrieben oder ersetzt. Endpunktrichtlinien steuern unabhängig vom Endpunkt den Zugriff auf den angegebenen Service.

Endpunktrichtlinien müssen im JSON-Format erstellt werden.

Weitere Informationen finden Sie unter [Steuerung des Zugriffs auf Services mit VPC-Endpunkten](#) im Amazon VPC User Guide.

Im Folgenden finden Sie ein Beispiel für eine Endpunktrichtlinie für CloudWatch Logs. Diese Richtlinie ermöglicht Benutzern, die über die VPC eine Verbindung zu CloudWatch Logs herstellen, Log-Streams zu erstellen und Logs an CloudWatch Logs zu senden, und verhindert, dass sie andere CloudWatch Logs-Aktionen ausführen.

```
{
  "Statement": [
    {
      "Sid": "PutOnly",
      "Principal": "*",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

So ändern Sie die VPC-Endpunktrichtlinie für Logs CloudWatch

1. Öffnen Sie die Amazon-VPC-Konsole unter <https://console.aws.amazon.com/vpc/>.

2. Wählen Sie im Navigationsbereich Endpunkte aus.
3. Wenn Sie den Endpunkt für CloudWatch Logs noch nicht erstellt haben, wählen Sie Create Endpoint aus. Wählen Sie dann com.amazonaws aus. *Region*.logs und wählen Sie Create Endpoint.
4. Wählen Sie com.amazonaws aus. *Region*.logs-Endpunkt und wählen Sie in der unteren Hälfte des Bildschirms die Registerkarte Richtlinie aus.
5. Wählen Sie Richtlinie bearbeiten und nehmen Sie die Änderungen an der Richtlinie vor.

Support für VPC-Kontextschlüssel

CloudWatch Logs unterstützt die Schlüssel `aws:SourceVpc` und die `aws:SourceVpce` Kontextschlüssel, die den Zugriff auf bestimmte VPCs oder spezifische VPC-Endpunkte einschränken können. Diese Schlüssel funktionieren nur, wenn der Benutzer VPC-Endpunkte verwendet. Weitere Informationen finden Sie unter [Schlüssel, die für manche Services verfügbar sind](#) im IAM-Benutzerhandbuch.

Logging CloudWatch protokolliert API- und Konsolenoperationen in AWS CloudTrail

Amazon CloudWatch Logs ist in einen Service integriert [AWS CloudTrail](#), der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem ausgeführten Aktionen bereitstellt AWS-Service. CloudTrail erfasst API-Aufrufe für CloudWatch Logs als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der CloudWatch Logs-Konsole und Codeaufrufen für die CloudWatch Logs-API-Operationen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an CloudWatch Logs gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, den Zeitpunkt der Anfrage und weitere Details ermitteln.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anforderung mit Anmeldeinformationen des Root-Benutzers oder des Benutzers gestellt wurde.
- Die Anforderung wurde im Namen eines IAM-Identity-Center-Benutzers erstellt.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde.

CloudTrail ist in Ihrem aktiv AWS-Konto , wenn Sie das Konto erstellen, und Sie haben automatisch Zugriff auf den CloudTrail Eventverlauf. Der CloudTrail Ereignisverlauf bietet eine einsehbare, durchsuchbare, herunterladbare und unveränderliche Aufzeichnung der aufgezeichneten Verwaltungsereignisse der letzten 90 Tage in einem. AWS-Region Weitere Informationen finden Sie im AWS CloudTrail Benutzerhandbuch unter [Arbeiten mit dem CloudTrail Ereignisverlauf](#). Für die Anzeige des Eventverlaufs CloudTrail fallen keine Gebühren an.

Für eine fortlaufende Aufzeichnung der Ereignisse in AWS-Konto den letzten 90 Tagen erstellen Sie einen Trail- oder [CloudTrailLake-Event-Datenspeicher](#).

CloudTrail Pfade

Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Alle mit dem erstellten Pfade AWS-Managementkonsole sind regionsübergreifend. Sie können mithilfe von AWS CLI einen Einzel-Region- oder einen Multi-Region-Trail erstellen. Es

wird empfohlen, einen Trail mit mehreren Regionen zu erstellen, da Sie alle Aktivitäten AWS-Regionen in Ihrem Konto erfassen. Wenn Sie einen Einzel-Region-Trail erstellen, können Sie nur die Ereignisse anzeigen, die im AWS-Region des Trails protokolliert wurden. Weitere Informationen zu Trails finden Sie unter [Erstellen eines Trails für Ihr AWS-Konto](#) und [Erstellen eines Trails für eine Organisation](#) im AWS CloudTrail -Benutzerhandbuch.

Sie können eine Kopie Ihrer laufenden Verwaltungsereignisse kostenlos an Ihren Amazon S3 S3-Bucket senden, CloudTrail indem Sie einen Trail erstellen. Es fallen jedoch Amazon S3 S3-Speichergebühren an. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter [AWS CloudTrail Preise](#). Informationen zu Amazon-S3-Preisen finden Sie unter [Amazon S3 – Preise](#).

CloudTrail Datenspeicher für Ereignisse in Lake

CloudTrail Mit Lake können Sie SQL-basierte Abfragen für Ihre Ereignisse ausführen. CloudTrail [Lake konvertiert bestehende Ereignisse im zeilenbasierten JSON-Format in das Apache ORC-Format](#). ORC ist ein spaltenförmiges Speicherformat, das für den schnellen Abruf von Daten optimiert ist. Die Ereignisse werden in Ereignisdatenspeichern zusammengefasst, bei denen es sich um unveränderliche Sammlungen von Ereignissen handelt, die auf Kriterien basieren, die Sie mit Hilfe von [erweiterten Ereignisselektoren](#) auswählen. Die Selektoren, die Sie auf einen Ereignisdatenspeicher anwenden, steuern, welche Ereignisse bestehen bleiben und für Sie zur Abfrage verfügbar sind. Weitere Informationen zu CloudTrail Lake finden Sie unter [Arbeiten mit AWS CloudTrail Lake](#) im AWS CloudTrail Benutzerhandbuch.

CloudTrail Für das Speichern und Abfragen von Ereignisdaten in Lake fallen Kosten an. Beim Erstellen eines Ereignisdatenspeichers wählen Sie die [Preisoption](#) aus, die für den Ereignisdatenspeicher genutzt werden soll. Die Preisoption bestimmt die Kosten für die Erfassung und Speicherung von Ereignissen sowie die standardmäßige und maximale Aufbewahrungsdauer für den Ereignisdatenspeicher. Weitere Informationen zur Preisgestaltung finden Sie unter CloudTrail [AWS CloudTrail Preisgestaltung](#).

CloudWatch Logs unterstützt die Protokollierung der folgenden Aktionen als Ereignisse in CloudTrail Protokolldateien:

- [AssociateKmsKey](#)
- [CancelExportTask](#)
- [CreateDelivery](#)
- [CreateExportTask](#)
- [CreateLogAnomalyDetector](#)

- [CreateLogGroup](#)
- [CreateLogStream](#)
- [DeleteAccountPolicy](#)
- [DeleteDataProtectionPolicy](#)
- [DeleteDelivery](#)
- [DeleteDeliveryDestination](#)
- [DeleteDeliveryDestinationPolicy](#)
- [DeleteDeliverySource](#)
- [DeleteDestination](#)
- [DeleteIndexPolicy](#)
- [DeleteIntegration](#)
- [DeleteLogAnomalyDetector](#)
- [DeleteLogGroup](#)
- [DeleteLogStream](#)
- [DeleteMetricFilter](#)
- [DeleteQueryDefinition](#)
- [DeleteResourcePolicy](#)
- [DeleteRetentionPolicy](#)
- [DeleteSubscriptionFilter](#)
- [DeleteTransformer](#)
- [DescribeAccountPolicies](#)
- [DescribeConfigurationTemplates](#)
- [DescribeDeliveries](#)
- [DescribeDeliveryDestinations](#)
- [DescribeDeliverySources](#)
- [DescribeDestinations](#)
- [DescribeExportTasks](#)
- [DescribeFieldIndexes](#)
- [DescribeIndexPolicies](#)

- [DescribeLogGroups](#)
- [DescribeLogStreams](#)
- [DescribeMetricFilters](#)
- [DescribeQueries](#)
- [DescribeQueryDefinitions](#)
- [DescribeResourcePolicies](#)
- [DescribeSubscriptionFilters](#)
- [DisassociateKmsKey](#)
- [FilterLogEvents](#)
- [GetDataProtectionPolicy](#)
- [GetDelivery](#)
- [GetDeliveryDestination](#)
- [GetDeliveryDestinationPolicy](#)
- [GetDeliverySource](#)
- [GetIntegration](#)
- [GetLogAnomalyDetector](#)
- [GetLogEvents](#)
- [GetLogGroupFields](#)
- [GetLogRecord](#)
- [GetQueryResults](#)
- [GetTransformer](#)
- [ListAnomalies](#)
- [ListIntegrations](#)
- [ListLogAnomalyDetectors](#)
- [ListLogGroups](#)
- [ListLogGroupsForQuery](#)
- [ListTagsForResource](#)
- [ListTagsLogGroup](#)
- [PutAccountPolicy](#)
- [PutDataProtectionPolicy](#)

- [PutDeliveryDestination](#)
- [PutDeliveryDestinationPolicy](#)
- [PutDeliverySource](#)
- [PutDestination](#)
- [PutDestinationPolicy](#)
- [PutIndexPolicy](#)
- [PutIntegration](#)
- [PutMetricFilter](#)
- [PutQueryDefinition](#)
- [PutResourcePolicy](#)
- [PutRetentionPolicy](#)
- [PutSubscriptionFilter](#)
- [PutTransformer](#)
- [StartLiveTail](#)
- [StartQuery](#)
- [StopQuery](#)
- [TagResource](#)
- [TestMetricFilter](#)
- [TestTransformer](#)
- [UntagResource](#)
- [UpdateAnomaly](#)
- [UpdateDeliveryConfiguration](#)
- [UpdateLogAnomalyDetector](#)

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Gibt an, ob die Anforderung mit Root- oder IAM-Benutzer-Anmeldeinformationen ausgeführt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter [CloudTrail userIdentity-Element](#).

Informationen zur Generierung von Abfragen in CloudTrail

CloudTrail Die Protokollierung von Ereignissen in der Query Generator-Konsole wird ebenfalls unterstützt. Der Abfragegenerator wird derzeit für CloudWatch Logs Insights und CloudWatch Metric Insights unterstützt. In diesen CloudTrail Fällen eventSource ist `dasmonitoring.amazonaws.com`.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `GenerateQuery`Aktion in CloudWatch Logs Insights demonstriert.

```
{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111222333444:role/Administrator",
        "accountId": "123456789012",
        "userName": "SAMPLE_NAME"
      },
      "attributes": {
        "creationDate": "2020-04-08T21:43:24Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2020-04-08T23:06:30Z",
  "eventSource": "monitoring.amazonaws.com",
  "eventName": "GenerateQuery",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "exampleUserAgent",
  "requestParameters": {
    "query_ask": "****",
```



```
    "query_type": "LogsInsights",
    "logs_insights": {
      "fields": "****",
      "log_group_names": ["yourloggroup"]
    },
    "include_description": true
  },
  "responseElements": null,
  "requestID": "2f56318c-cfbd-4b60-9d93-1234567890",
  "eventID": "52723fd9-4a54-478c-ac55-1234567890",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "eventCategory": "Management"
}
```

Grundlagen zu -Protokolldateieinträgen

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Der folgende Eintrag in der Protokolldatei zeigt, dass ein Benutzer die CreateExportTaskAktion CloudWatch Logs aufgerufen hat.

```
{
  "eventVersion": "1.03",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:user/someuser",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "someuser"
  },
  "eventTime": "2016-02-08T06:35:14Z",
  "eventSource": "logs.amazonaws.com",
```

```
"eventName": "CreateExportTask",
"awsRegion": "us-east-1",
"sourceIPAddress": "127.0.0.1",
"userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
"requestParameters": {
  "destination": "yourdestination",
  "logGroupName": "yourloggroup",
  "to": 123456789012,
  "from": 0,
  "taskName": "yourtask"
},
"responseElements": {
  "taskId": "15e5e534-9548-44ab-a221-64d9d2b27b9b"
},
"requestID": "1cd74c1c-ce2e-12e6-99a9-8dbb26bd06c9",
"eventID": "fd072859-bd7c-4865-9e76-8e364e89307c",
"eventType": "AwsApiCall",
"apiVersion": "20140328",
"recipientAccountId": "123456789012"
}
```

Überwachung mit CloudWatch Metriken

Sie können die Tabellen in diesem Abschnitt verwenden, um die Metriken zu überprüfen, die Amazon CloudWatch Logs CloudWatch jede Minute an Amazon sendet.

CloudWatch Protokolliert Metriken

Der AWS/Logs-Namespace enthält die folgenden Metriken.

Metrik	Description
CallCount	Die Anzahl der angegebenen API-Operationen, die in Ihrem Konto ausgeführt werden. CallCount ist eine Metrik zur Nutzung des CloudWatch Logs-Dienstes. Weitere Informationen finden Sie unter CloudWatch Protokolliert Metriken zur Dienstnutzung. Gültige Dimensionen: Klasse, Ressource, Service, Typ Gültige Statistiken: Summe Einheiten: keine
DeliveryErrors	Die Anzahl der Protokollereignisse, für die CloudWatch Logs beim Weiterleiten von Daten an das Abonnementziel einen Fehler gemeldet hat. Wenn der Zieldienst einen Fehler zurückgibt, der wiederholt werden kann, z. B. eine Drosselungsausnahme oder eine Dienstausnahme, die erneut versucht werden kann (z. B. HTTP 5xx), wiederholt CloudWatch Logs die Zustellung für bis zu 24 Stunden. CloudWatch Logs versucht nicht, erneut zuzustellen, wenn es sich bei dem Fehler um einen Fehler handelt, der nicht wiederholt werden kann, wie z. B. oder. <code>AccessDeniedException</code> <code>ResourceNotFoundException</code> Gültige Abmessungen:,,, LogGroupName DestinationType FilterName PolicyLevel Gültige Statistiken: Summe

Metrik	Description
	Einheiten: keine
DeliveryThrottling	Die Anzahl der Protokollereignisse, für die CloudWatch Logs bei der Weiterleitung von Daten an das Abonnementziel gedrosselt wurde. Wenn der Zieldienst einen Fehler zurückgibt, der wiederholt werden kann, z. B. eine Drosselungs Ausnahme oder eine Dienstaufnahme, die erneut versucht werden kann (z. B. HTTP 5xx), wiederholt CloudWatch Logs die Zustellung für bis zu 24 Stunden. CloudWatch Logs versucht nicht, erneut zuzustellen, wenn es sich bei dem Fehler um einen Fehler handelt, der nicht wiederholt werden kann, wie z. B. oder. <code>AccessDeniedException</code> <code>ResourceNotFoundException</code> Gültige Abmessungen:,,, LogGroupName DestinationType FilterName PolicyLevel Gültige Statistiken: Summe Einheiten: keine
EMFDisabledErrors	Die Anzahl der Protokollereignisse im EMF-Format, die aufgrund einer aktiven Kontorichtlinie vom Typ METRIC_EXTRACTION_POLICY, die der Protokollgruppe entspricht, ignoriert wurden. Weitere Informationen zu den Metrikextraktionsrichtlinien finden Sie unter. PutAccountPolicy Gültige Dimensionen: LogGroupName Gültige Statistiken: Summe Einheiten: keine

Metrik	Description
EMFParsingErrors	Die Anzahl der Analysefehler, die bei der Verarbeitung von EMF (eingebettetes Metrikformat)-Protokollen aufgetreten sind. Solche Fehler treten auf, wenn Protokolle als eingebettetes Metrikformat identifiziert werden, aber nicht dem richtigen Format folgen. Weitere Informationen zum eingebetteten Metrikformat finden Sie unter Spezifikation: Eingebettetes Metrikformat. Gültige Dimensionen: LogGroupName Gültige Statistiken: Summe Einheiten: keine
EMFValidationErrors	Die Anzahl der Validierungsfehler, die bei der Verarbeitung von EMF (eingebettetes Metrikformat)-Protokollen aufgetreten sind. Diese Fehler treten auf, wenn Metrikdefinitionen in EMF-Protokollen nicht den EMF- und MetricDatum -Spezifikationen entsprechen. Informationen zum eingebetteten metrischen Format finden Sie unter Spezifikation: Eingebettetes metrisches CloudWatch Format. Informationen zum Datentyp MetricDatum finden Sie MetricDatum in der Amazon CloudWatch API-Referenz.  Note Bestimmte Validierungsfehler können dazu führen, dass mehrere Metriken innerhalb eines EMF-Protokolls nicht veröffentlicht werden. Beispielsweise werden alle Metriken, die mit einem ungültigen Namespace festgelegt wurden, gelöscht. Gültige Dimensionen: LogGroupName Gültige Statistiken: Summe Einheiten: keine

Metrik	Description
ErrorCount	Die Anzahl der API-Operationen, die in Ihrem Konto ausgeführt wurden und zu Fehlern geführt haben. ErrorCount ist eine Metrik zur Nutzung des CloudWatch Logs-Service. Weitere Informationen finden Sie unter CloudWatch Protokolliert Metriken zur Dienstnutzung. Gültige Dimensionen: Klasse, Ressource, Service, Typ Gültige Statistiken: Summe Einheiten: keine
ForwardedBytes	Die Menge der Protokollereignisse in komprimierten Byte, die zum Abonnementziel weitergeleitet wurden. Gültige Abmessungen: LogGroupName, DestinationType, FilterName Gültige Statistiken: Summe Einheiten: Byte
Forwarded LogEvents	Die Zahl der Protokollereignisse, die zum Abonnementziel weitergeleitet wurden. Gültige Abmessungen: LogGroupName, DestinationType, FilterName, PolicyLevel Gültige Statistiken: Summe Einheiten: keine

Metrik	Description
IncomingBytes	Das Volumen der Protokollereignisse in unkomprimierten Byte, die in CloudWatch Logs hochgeladen wurden. In Verbindung mit der Dimension LogGroupName ist dies die Menge der Protokollereignisse in nicht komprimierten Byte, die in die Protokollgruppe hochgeladen wurden. Gültige Abmessungen: LogGroupName Gültige Statistiken: Summe Einheiten: Byte
IncomingLogEvents	Die Anzahl der Protokollereignisse, die in CloudWatch Logs hochgeladen wurden. In Verbindung mit der Dimension LogGroupName ist dies die Anzahl der Protokollereignisse, die in die Protokollgruppe hochgeladen wurden. Gültige Abmessungen: LogGroupName Gültige Statistiken: Summe Einheiten: keine
LogEventsWithFindings	Die Anzahl der Protokollereignisse, die mit einer Datenzeichenfolge übereinstimmten, die Sie mit der Datenschutzfunktion CloudWatch Logs überprüfen. Weitere Informationen finden Sie unter Den Schutz vertraulicher Protokolldaten mit Maskierung unterstützen. Gültige Dimensionen: keine Gültige Statistiken: Summe Einheiten: keine

Metrik	Description
ThrottleCount	Die Anzahl der API-Operationen, die in Ihrem Konto ausgeführt wurden, die aufgrund von Nutzungskontingenten gedrosselt wurden. ThrottleCount ist eine Metrik zur Nutzung des CloudWatch Logs-Dienstes. Weitere Informationen finden Sie unter CloudWatch Protokolliert Metriken zur Dienstnutzung. Gültige Dimensionen: Klasse, Ressource, Service, Typ Gültige Statistiken: Summe Einheiten: keine

Dimensionen für CloudWatch Logs-Metriken

Die Dimensionen, die Sie für die meisten CloudWatch Logs-Metriken verwenden können, sind in der folgenden Tabelle aufgeführt.

Dimension	Description
LogGroupName	Der Name der CloudWatch Logs-Protokollgruppe, für die Metriken angezeigt werden sollen.
DestinationType	Das Abonnementziel für die CloudWatch Logs-Daten. Dabei kann es sich AWS Lambda um Amazon Kinesis Data Streams oder Amazon Data Firehose handeln.
FilterName	Der Name des Abonnementfilters, der Daten aus der Protokollgruppe an das Ziel weiterleitet. Der Name des Abonnementsfilters wird automatisch in ASCII umgewandelt und alle nicht unterstützten Zeichen werden durch ein Fragezeichen (?) ersetzt. CloudWatch

Metrische Abmessungen des Abonnementfilters

Die Dimensionen für Metriken, die sich auf Abonnementfilter auf Kontoebene beziehen, sind in der folgenden Tabelle aufgeführt.

Dimension	Description
PolicyLevel	Die Ebene, auf der die Richtlinie gilt. Derzeit ist der einzig gültige Wert für diese Dimension AccountPolicy
DestinationType	Das Abonnementziel für die CloudWatch Logs-Daten. Dabei kann es sich AWS Lambda um Amazon Kinesis Data Streams oder Amazon Data Firehose handeln.
FilterName	Der Name des Abonnementfilters, der Daten aus der Protokollgruppe an das Ziel weiterleitet. Der Name des Abonnementsfilters wird automatisch in ASCII umgewandelt und alle nicht unterstützten Zeichen werden durch ein Fragezeichen (?) ersetzt. CloudWatch

Metriken und Dimensionen von Log Transformer

CloudWatch Logs veröffentlicht die folgenden Log-Transformer-Metriken CloudWatch im AWS/Logs Namespace.

Metrik	Description
TransformationErrors	Die Anzahl der Fehler, die bei der Transformation von Protokollereignissen mit dem angegebenen Transformer aufgetreten sind. Einheit: keine Gültige Statistiken: Summe
TransformedBytes	Das Volumen der Ausgabe transformierter Protokollereignisse in unkomprimierten Byte. Einheit: Byte

Metrik	Description
	Gültige Statistiken: Summe
TransformedLogEvents	Die Anzahl der transformierten Protokollereignisse. Einheit: keine Gültige Statistiken: Summe

Die folgenden Dimensionen werden von Transformer-Metriken verwendet.

Dimension	Description
LogGroupName	Diese Dimension wird nur für log-group-level Transformatoren verwendet.
PolicyLevel	Diese Dimension wird nur für Transformatoren auf Kontoebene verwendet. Derzeit ist der einzig gültige Wert für diese Dimension AccountPolicy

Metriken und Dimensionen der Zentralisierung

Für eine zentralisierte Überwachung über mehrere Konten und Regionen hinweg können Sie CloudWatch Logs Centralization verwenden, um Protokolldaten und Metriken an einem zentralen Ort zu konsolidieren. Weitere Informationen finden Sie unter [Cross-account Regionsübergreifende Protokollzentralisierung](#).

CloudWatch Logs veröffentlicht die folgenden Zentralisierungsmetriken CloudWatch im AWS/Logs Namespace. Diese Metriken helfen Ihnen bei der Überwachung der Replikation von Protokolldaten von Quellkonten zu Zielkonten, wenn Sie Regeln zur Zentralisierung von CloudWatch Logs verwenden.

Note

CloudWatch beginnt kurz nachdem Sie eine Zentralisierungsregel erstellt haben, mit der Berichterstattung über Zentralisierungsmetriken. Metriken werden nach bestem Wissen

veröffentlicht und erfassen nur neue Protokollereignisse, die nach der Erstellung der Zentralisierungsregel eintreffen.

Metrik	Description	Veröffentlicht in
IncomingCopiedBytes	Das Volumen der Protokolldaten in unkomprimierten Byte, das in das Zielkonto repliziert wurde. Diese Metrik gilt nur für neue Protokollereignisse, die nach der Erstellung der Zentralisierungsregel eintreffen. Gültige Abmessungen: SourceRegion, SourceAccount Gültige Statistiken: Summe Einheiten: Byte	Zielkonto
IncomingCopiedLogEvents	Die Anzahl der Protokollereignisse, die in das Zielkonto repliziert wurden. Diese Metrik gilt nur für neue Protokollereignisse, die nach der Erstellung der Zentralisierungsregel eintreffen. Gültige Abmessungen: SourceRegion, SourceAccount Gültige Statistiken: Summe Einheiten: keine	Zielkonto
OutgoingCopiedBytes	Das Volumen der Protokolldaten in unkomprimierten Byte, das von Quellkonten an das Zielkonto gesendet wurde. Diese Metrik gilt nur für neue Protokollereignisse, die nach der Erstellung der Zentralisierungsregel an das Zielkonto kopiert werden. Gültige Dimensionen: DestinationRegion	Quellkonto

Metrik	Description	Veröffentlicht in
	Gültige Statistiken: Summe Einheiten: Byte	
OutgoingCopiedLogEvents	Die Anzahl der Protokollereignisse, die von Quellkonten an das Zielkonto gesendet wurden. Diese Metrik gilt nur für neue Protokollereignisse, die nach der Erstellung der Zentralisierungsregel an das Ziel kopiert werden. Gültige Dimensionen: DestinationRegion Gültige Statistiken: Summe Einheiten: keine	Quellkonto
CentralizationError	Die Anzahl der Fehler, die beim Replizieren von Protokolldaten aufgetreten sind. Fehler können aus verschiedenen Gründen auftreten, z. B. aufgrund von Problemen mit KMS-Schlüsselberechtigungen, Kontingenzbeschränkungen für Protokollgruppen oder nicht übereinstimmende Protokollebenen. Überwachen Sie die <code>ErrorType</code> Dimension, um bestimmte Protokollgruppen oder Protokolldatenströme, bei denen die Replikation fehlgeschlagen ist, und deren Fehlerursachen zu identifizieren. Gültige Dimensionen: ErrorType Gültige Statistiken: Summe Einheiten: keine	Zielkonto

Metrik	Description	Veröffentlicht in
CentralizationThrottled	Wie oft die Zentralisierungsverarbeitung gedrosselt wurde. Die Drosselung führt zu einer langsameren Verarbeitung der Zentralisierung, verhindert jedoch nicht, dass Protokolldaten repliziert werden. Gültige Abmessungen: DestinationRegion Gültige Statistiken: Summe Einheiten: keine	Quellkonto

Die folgenden Dimensionen werden für Zentralisierungsmetriken verwendet.

Dimension	Description
SourceRegion	Die AWS Region, aus der die Quellprotokolldaten stammen.
SourceAccount	Die AWS Konto-ID, aus der die Quellprotokolldaten stammen.
DestinationRegion	Die AWS Region, in die die Protokolldaten repliziert werden.
ErrorType	Die Art des Fehlers, der bei der Zentralisierung aufgetreten ist. Mögliche Werte sind: - LogGroupQuotaExceeded : Das Zielkonto hat sein Protokollgruppenkontingent erreicht. - InvalidKMS : Der KMS-Schlüssel fehlt, wurde gelöscht, ist nicht aktiviert oder er ist asymmetrisch statt symmetrisch. - AccessDenied : Beim Versuch, Protokolldaten zu replizieren, wurde der Zugriff verweigert. Dies kann auf unzureichende Berechtigungen zurückzuführen sein, z. B. auf falsche KMS-Schlüsselberechtigungen, fehlende IAM-Berechtigungen oder restriktive Ressourcenrichtlinien.

Dimension	Description
	• LogTierMismatch : Die Protokollebene der Quell- und Zielprotokollgruppen stimmen nicht überein. • InvalidLogStream : Beim Erstellen oder Schreiben in einen Protokollstream wurde ein ungültiger Parameter gefunden, z. B. wenn der Name des Protokolldatenstroms die maximale Länge überschreitet. • InvalidLogGroup : Beim Erstellen oder Konfigurieren einer Protokollgruppe im Zielkonto wurde ein ungültiger Parameter gefunden. • DestinationEncryptionMismatch : Die Zielprotokollgruppe ist bereits mit einer anderen KMS-Verschlüsselungskonfiguration als der in der Zentralisierungsregel angegebenen vorhanden.

CloudWatch Protokolliert Metriken zur Dienstnutzung

CloudWatch Logs sendet Metriken CloudWatch , die die Nutzung von CloudWatch Logs-API-Vorgängen verfolgen. Diese Metriken entsprechen den AWS Servicekontingenten. Die Verfolgung dieser Metriken kann Ihnen dabei helfen, Ihre Kontingente proaktiv zu verwalten. Weitere Informationen finden Sie unter [Service Quotas – Integration und Nutzungsmetriken](#).

So können Sie beispielsweise die **ThrottleCount**-Metrik verfolgen oder einen Alarm für diese Metrik einstellen. Wenn der Wert dieser Metrik steigt, sollten Sie eine Kontingenterhöhung für die API-Operation anfordern, die gedrosselt wird. Weitere Informationen zu den Dienstkontingenten für CloudWatch Logs finden Sie unter [CloudWatch Protokolliert Kontingente](#).

CloudWatch Logs veröffentlicht jede Minute Messdaten zur Nutzung von Dienstkontingenten in den AWS/Logs Namespaces AWS/Usage und.

In der folgenden Tabelle sind die von Logs veröffentlichten Messwerte zur Servicenutzung aufgeführt. CloudWatch Diese Metriken hat keine angegebene Einheit. Die nützlichste Statistik für diese Metriken ist SUM, die die Gesamtanzahl der Operationen für den 1-Minuten-Zeitraum darstellt.

Jede dieser Metriken wird mit Werten für alle Service-, Class-, Type-, und Resource-Dimensionen veröffentlicht. Sie werden auch mit einer einzigen Dimension namens Account

Metrics veröffentlicht. Verwenden Sie die **Account Metrics**-Dimension, um die Summe der Metriken für alle API-Operationen in Ihrem Konto anzuzeigen. Verwenden Sie die anderen Dimensionen, und geben Sie den Namen einer API-Operation für die **Resource**-Dimension an, um die Metriken für diese bestimmte API zu finden.

Metriken

Metrik	Description
CallCount	Die Anzahl der angegebenen Operationen, die in Ihrem Konto ausgeführt werden. CallCount wird sowohl in der AWS/Usage und AWS/Logs-Namespaces veröffentlicht.
ErrorCount	Die Anzahl der API-Operationen, die in Ihrem Konto ausgeführt wurden und zu Fehlern geführt haben. ErrorCount wird nur in der AWS/Logs veröffentlicht.
ThrottleCount	Die Anzahl der API-Operationen, die in Ihrem Konto ausgeführt wurden, die aufgrund von Nutzungskontingenten gedrosselt wurden. ThrottleCount wird nur in der AWS/Logs veröffentlicht.

Dimensions (Abmessungen)

Dimension	Description
Account metrics	Verwenden Sie diese Dimension, um eine Summe der Metrik in allen CloudWatch Protokollen abzurufen APIs. Wenn Sie die Metriken für eine bestimmte API anzeigen möchten, verwenden Sie die anderen Dimensionen, die in dieser Tabelle aufgeführt sind, und geben Sie den API-Namen als Wert von Resource an.

Dimension	Description
Service	Der Name des AWS Dienstes, der die Ressource enthält. Für Metriken zur Nutzung von CloudWatch Logs lautet der Wert für diese Dimension Logs.
Class	Die Klasse der Ressource, die verfolgt wird. CloudWatch Die Metriken zur Nutzung der Logs API verwenden diese Dimension mit einem Wert vonNone.
Type	Der Typ der nachverfolgten Ressource. Wenn die Service-Dimension Logs ist, ist API der derzeit einzige gültige Wert für Type.
Resource	Der Name der API-Operation. Gültige Werte umfassen alle API-Operationsnamen, die in Aktionen aufgelistet sind. Zum Beispiel PutLogEvents

CloudWatch Protokolliert Kontingente

Sie können die Tabelle in diesem Abschnitt verwenden, um die Standard-Servicekontingente, auch Limits genannt, für ein AWS Konto in Amazon CloudWatch Logs zu überprüfen. Die meisten Service Quotas, aber nicht alle, sind unter dem Amazon CloudWatch Logs-Namespace in der Service Quotas-Konsole aufgeführt.

Note

Wenn Sie eine Erhöhung des Kontingents für eines dieser Kontingente beantragen möchten, sehen Sie sich [das Verfahren](#) in diesem Abschnitt an.

Name	Standard	Anpas	Description
Aktive Exportaufgabe	Jede unterstützte Region: 1	Nein	Die Anzahl der aktiven (laufenden oder ausstehenden) Exportaufgaben pro Konto
AssociateKmsKey Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von assoziierten KMS-Key-Aufrufen pro Sekunde pro Region account/per
AssociateSourceToS3TableIntegration Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5	Nein	Die maximale Anzahl von AssociateSourceToS3TableIntegration Aufrufen pro Sekunde pro account/per Region
AssociateSourceToS3TableIntegration Drosselt das Limit für Transaktionen pro Sekunde in einem Burst	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von AssociateSourceToS3TableIntegration Aufrufen pro Sekunde pro account/per Region in einem Burst

Name	Standard	Anpas	Description
Batch-Größe	Jede unterstützte Region: 1 Megabyte	Nein	Die maximale Batchgröße einer Anforderung „put-log-events“ in MB
CancelExportTask Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Aufrufen von CancelExport-Aufgaben pro Sekunde pro Region account/per
CancelImportTask Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von CancelImportTask Aufrufen pro Sekunde pro account/per Region
CancelImportTask Drosselt das Limit für Transaktionen pro Sekunde in einem Burst	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von CancelImportTask Aufrufen pro Sekunde pro account/per Region in einem Burst
CreateExportTask Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Aufrufen von CreateExport-Aufgaben pro Sekunde pro Region account/per
CreateImportTask Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von CreateImportTask Aufrufen pro Sekunde pro account/per Region

Name	Standard	Anpas	Description
CreateImportTask Drosselt das Limit für Transaktionen pro Sekunde in einem Burst	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von CreateImportTask Aufrufen pro Sekunde pro account/per Region in einem Burst
CreateLogGroup Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 10 pro Sekunde	Ja	Die maximale Anzahl von Create-Log-Group-Aufrufen pro Sekunde pro Region account/per
CreateLogStream Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 50 pro Sekunde	Ja	Die maximale Anzahl von Create-Log-Stream-Aufrufen pro Sekunde und Region account/per
Datenarchivierung	Jede unterstützte Region: 5 Gigabyte	Nein	Die maximale Menge der kostenlosen Datenarchivierung in GB
DeleteDataProtectionPolicy Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5	Nein	Die maximale Anzahl von Aufrufen zum Löschen von Datenschutzrichtlinien pro Sekunde und Region account/per
DeleteDestination Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Aufrufen zum Löschen von Zielen pro Sekunde pro Region account/per
DeleteLogGroup Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 10 pro Sekunde	Ja	Die maximale Anzahl von Gruppenaufrufen zum Löschen von Protokollen pro Sekunde pro Region account/per

Name	Standard	Anpas	Description
DeleteLogStream Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 15 pro Sekunde	Ja	Die maximale Anzahl von Delete-Log-Stream-Aufrufen pro Sekunde und Region account/per
DeleteMetricFilter Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Delete-Metric-Filter-Aufrufen pro Sekunde und Region account/per
DeleteRetentionPolicy Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Aufrufen zur Lösch-/Aufbewahrungsrichtlinie pro Sekunde pro Region account/per
DeleteSubscriptionFilter Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Aufrufen zum Löschen von Abonnementfiltern pro Sekunde und Region account/per
DeleteTransformer Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Delete-Transformer-Aufrufen pro Sekunde und Region account/per
DescribeDestinations Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Describe-Destinations-Anrufen pro Sekunde pro Region account/per
DescribeExportTasks Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Describe-Export-Tasks-Aufrufen pro Sekunde und Region account/per

Name	Standard	Anpas	Description
DescribeImportTaskBatches Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von DescribeImportTask Batches Aufrufen pro Sekunde pro account/per Region
DescribeImportTaskBatches Drosselt das Limit für Transaktionen pro Sekunde in einem Burst	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von DescribeImportTask Batches Aufrufen pro Sekunde pro account/per Region in einem Burst
DescribeImportTasks Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von DescribeImportTasks Aufrufen pro Sekunde pro account/per Region
DescribeImportTasks Drosselt das Limit für Transaktionen pro Sekunde in einem Burst	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von DescribeImportTasks Aufrufen pro Sekunde pro account/per Region in einem Burst
DescribeLogGroups Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 10 pro Sekunde	Ja	Die maximale Anzahl von Describe-Log-Groups-Aufrufen pro Sekunde pro Region account/per
DescribeLogStreams Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 25 pro Sekunde	Ja	Die maximale Anzahl von Describe-Log-Streams-Aufrufen pro Sekunde und Region account/per

Name	Standard	Anpas	Description
DescribeMetricFilters Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Describe-Metric-Filter-Aufrufen pro Sekunde und Region account/per
DescribeSubscriptionFilters Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Aufrufen von describe-subscription-filters pro Sekunde und Region account/per
DisassociateSourceFromS3TableIntegration Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5	Nein	Die maximale Anzahl von DisassociateSourceFromS3TableIntegration Aufrufen pro Sekunde pro account/per Region
DisassociateSourceFromS3TableIntegration Drosselt das Limit für Transaktionen pro Sekunde in einem Burst	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von DisassociateSourceFromS3TableIntegration Aufrufen pro Sekunde pro account/per Region in einem Burst
Ereignis-Größe	Jede unterstützte Region: 1 024 Kilobyte	Nein	Die maximale Größe des Protokollereignisses in KB

Name	Standard	Anpas	Description
FilterLogEvents Drosselungslimit für Transaktionen pro Sekunde	us-east-1: 25 pro Sekunde ap-northeast-3: 5 pro Sekunde ap-southeast-3: 5 pro Sekunde ca-west-1: 5 pro Sekunde eu-central-1: 5 pro Sekunde il-central-1: 5 pro Sekunde Jede der anderen unterstützten Regionen: 10 pro Sekunde	Nein	Die maximale Anzahl von Aufrufen von Filter-Log-Events pro Sekunde pro Region account/per
GetDataProtectionPolicy Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5	Nein	Die maximale Anzahl von Get-Data-Protection-Policy-Aufrufen pro Sekunde und Region account/per

Name	Standard	Anpas	Description
GetLogEvents Drosselungslimit für Transaktionen pro Sekunde	us-west-2: 10 pro Sekunde ap-northeast-3: 10 pro Sekunde ap-southeast-3: 10 pro Sekunde ca-west-1: 10 pro Sekunde eu-central-1: 10 pro Sekunde eu-west-1: 10 pro Sekunde eu-west-3: 30 pro Sekunde il-central-1: 10 pro Sekunde Jede der anderen unterstützten Regionen: 25 pro Sekunde	Nein	Die maximale Anzahl von Get-Log-Events-Aufrufen pro Sekunde und Region account/per
GetQueryResults Drosselungslimit für Transaktionen pro Sekunde	me-central-1: 6 Jede der anderen unterstützten Regionen: 10	Nein	Die maximale Anzahl von Get-Query-Results-Aufrufen pro Sekunde und Region account/per

Name	Standard	Anpas	Description
GetTransformer Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Get-Transformer-Aufrufen pro Sekunde pro Region account/per
ListSourcesForS3TableIntegration Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5	Nein	Die maximale Anzahl von ListSourcesForS3TableIntegration Aufrufen pro Sekunde pro account/per Region
ListSourcesForS3TableIntegration Drosselt das Limit für Transaktionen pro Sekunde in einem Burst	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von ListSourcesForS3TableIntegration Aufrufen pro Sekunde pro account/per Region in einem Burst
ListTagsForResource Drosselungslimit für Transaktionen pro Sekunde	ca-central-1:30 pro Sekunde me-central-1:5 pro Sekunde Jede der anderen unterstützten Regionen: 10 pro Sekunde	Nein	Die maximale Anzahl von Aufrufen von List-Tags für Ressourcen pro Sekunde pro Region account/per
ListTagsLogGroup Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von List-Tags-Log-Group-Aufrufen pro Sekunde pro Region account/per
Limit für gleichzeitige Live-Tail-Sitzungen	Jede unterstützte Region: 15	Ja	Die maximale Anzahl gleichzeitiger aktiver Live Tail-Sitzungen pro Konto

Name	Standard	Anpas	Description
Protokollgruppen	Jede unterstützte Region: 1 000 000	Ja	Die maximale Anzahl von Protokollgruppen, die ein Konto haben kann
Protokollgruppen, die pro Live Tail-Sitzung gescannt wurden	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von Protokollgruppen, die pro Live Tail-Sitzung gescannt werden können
Metrikfilter pro Protokollgruppe	Jede unterstützte Region: 100	Nein	Die Anzahl der metrischen Filter pro Protokollgruppe
PutBearerTokenAuthentication Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5	Nein	Die maximale Anzahl von Put-Bearer-Token-Authentifizierungsaufrufen pro Sekunde und Region account/per
PutBearerTokenAuthentication Drosselungslimit für Transaktionen pro Sekunde in einem Burst	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von Put-Bearer-Token-Authentifizierungsaufrufen pro Sekunde und Region in einem Burst account/per
PutDataProtectionPolicy Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5	Nein	Die maximale Anzahl von Put-Data-Protection-Policy-Aufrufen pro Sekunde und Region account/per
PutDestination Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Put-Destination-Aufrufen pro Sekunde pro Region account/per

Name	Standard	Anpas	Description
PutDestinationPolicy Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Put-Destination-Policy-Aufrufen pro Sekunde und Region account/per
PutLogEvents Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 000 pro Sekunde	Ja	Die maximale Anzahl von Put-Log-Events-Aufrufen pro Sekunde und Region account/per
PutMetricFilter Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Put-Metric-Filter-Aufrufen pro Sekunde und Region account/per
PutRetentionPolicy Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Put-Retention-Policy-Aufrufen pro Sekunde und Region account/per
PutSubscriptionFilter Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Put-Subscription-Filter-Aufrufen pro Sekunde und Region account/per
PutTransformer Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Put-Transformer-Aufrufen pro Sekunde pro Region account/per
Ressourcenrichtlinien	Jede unterstützte Region: 10	Nein	Die maximale Anzahl von Ressourcenrichtlinien pro Region account/per

Name	Standard	Anpas	Description
StartLiveTail Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5	Nein	Die maximale Anzahl von Start-Live-Tail-Anrufen pro Sekunde und Region. account/per Dieses Limit gilt unabhängig sowohl für die Konsole als auch für die API
StartQuery Drosselungslimit für Transaktionen pro Sekunde	me-central-1:6 Jede der anderen unterstützten Regionen: 10	Nein	Die maximale Anzahl von Startabfrageaufrufen pro Sekunde pro Region account/per
Abonnementfilter pro Protokollgruppe	Jede unterstützte Region: 2	Nein	Die Anzahl der Abonnementfilter pro Protokollgruppe
TagLogGroup Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Tag-Log-Gruppenaufrufen pro Sekunde pro Region account/per
TagResource Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Tag-Resource-Aufrufen pro Sekunde pro Region account/per
TestMetricFilter Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Aufrufen von Testmetri kfiltern pro Sekunde und Region account/per

Name	Standard	Anpas	Description
TestTransformer Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Test-Transformer-Aufrufen pro Sekunde pro Region account/per
UntagLogGroup Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Untag-Log-Group-Aufrufen pro Sekunde pro Region account/per
UntagResource Drosselungslimit für Transaktionen pro Sekunde	Jede unterstützte Region: 5 pro Sekunde	Nein	Die maximale Anzahl von Untag-Resource-Aufrufen pro Sekunde und Region account/per
protokolliert Anomaliedetektoren	Jede unterstützte Region: 500	Ja	Die maximale Anzahl von aktiven Protokollanomaliedetektoren

Verwaltung Ihrer CloudWatch Logs-Dienstkontingente

CloudWatch Logs wurde in Service Quotas integriert, einen AWS Dienst, mit dem Sie Ihre Kontingente von einem zentralen Ort aus einsehen und verwalten können. Weitere Informationen zu Service Quotas finden Sie unter [Was sind Service Quotas](#) im Benutzerhandbuch für Service Quotas.

Mit Service Quotas können Sie ganz einfach den Wert Ihrer CloudWatch Logs-Dienstkontingente nachschlagen.

AWS-Managementkonsole

So zeigen Sie die Kontingente für den CloudWatch Logs-Service mit der Konsole an

1. Öffnen Sie die Service Quotas-Konsole unter <https://console.aws.amazon.com/servicequotas/>.
2. Wählen Sie im Navigationsbereich AWS -Services.
3. Suchen Sie in der AWS Serviceliste nach Amazon CloudWatch Logs und wählen Sie es aus.

In der Liste Service Quotas wird der Name der Service Quota, der angewendete Wert (falls verfügbar) und das AWS -Standardkontingent angezeigt. Zudem wird angezeigt, ob der Kontingentwert anpassbar ist.

4. Wählen Sie den Kontingentnamen, um zusätzliche Informationen zu einem Service Quota anzuzeigen, z. B. seine Beschreibung.
5. (Optional) Um eine Kontingenterhöhung zu beantragen, wählen Sie das Kontingent, das Sie erhöhen möchten, und dann Request quota increase (Kontingenterhöhung beantragen) aus, geben Sie die erforderlichen Informationen ein, und wählen Sie dann Request (Beantragen) aus.

Weitere Informationen zur Verwendung der Konsole mit Service Quotas finden Sie im [Benutzerhandbuch zu Service Quotas](#). Informationen zur Erhöhung eines Kontingents finden Sie unter [Anfordern einer Kontingenterhöhung](#) im Benutzerhandbuch zu Service Quotas.

AWS CLI

Um die Service-Kontingente für CloudWatch Logs anzuzeigen, verwenden Sie den AWS CLI

Führen Sie den folgenden Befehl aus, um die standardmäßigen CloudWatch Logs-Kontingente anzuzeigen.

```
aws service-quotas list-aws-default-service-quotas \
  --query 'Quotas[*]'.
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
  --service-code logs \
  --output table
```

Weitere Informationen zur Verwendung von Service Quotas finden Sie in der AWS CLI [AWS CLI Befehlsreferenz für Dienstkontingente](#). Informationen zum Beantragen einer Kontingenterhöhung finden Sie unter dem [request-service-quota-increase](#)-Befehl in der [AWS CLI - Befehlsreferenz](#).

Dokumentverlauf

In der folgenden Tabelle werden wichtige Änderungen in den einzelnen Versionen des CloudWatch Logs-Benutzerhandbuchs ab Juni 2018 beschrieben. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Die Richtlinie für AWSServiceRoleForLogDelivery dienstbezogene Rollen wurde aktualisiert	CloudWatch Es wurden Protokolle kms:GenerateDataKey und kms:Decrypt Berechtigungen für die Richtlinie für AWSServiceRoleForLogDelivery dienstverknüpfte Rollen hinzugefügt. Diese Berechtigungen ermöglichen die Protokollzustellung an Firehose-Lieferdatenströme, die serverseitige Verschlüsselung mit vom Kunden verwalteten Schlüsseln (SSE-CMK) verwenden. Weitere Informationen finden Sie unter CloudWatch Protokollierung von Aktualisierungen von AWS dienstverknüpften Rollen .	15. Mai 2026
Gespeicherte Abfragen mit Parametern für CloudWatch Logs Insights	In CloudWatch Logs Insights können Sie jetzt wiederverwendbare Abfragevorlagen mit benannten Parametern erstellen. Definieren Sie eine einzelne Vorlage und übergeben Sie zur Laufzeit unterschiedliche Werte, anstatt	26. März 2026

mehrere nahezu identische gespeicherte Abfragen zu verwalten. Weitere Informationen finden Sie unter [Gespeicherte Abfragen mit Parametern verwenden](#).

[Neue Richtlinie zur Verwaltung von CloudWatch von Protokollen CloudWatchLogsAPIKeyAccess](#)

CloudWatch Logs hat eine neue verwaltete Richtlinie veröffentlicht `CloudWatchLogsAPIKeyAccess`, die die Authentifizierung mit CloudWatch Logs-API-Schlüsseln und die verschlüsselte Protokollaufnahme ermöglicht. Weitere Informationen finden Sie unter [CloudWatch Protokollierung von Aktualisierungen AWS verwalteter Richtlinien](#).

17. Februar 2026

[Aktualisierte Richtlinie CloudWatchLogsReadOnlyAccess](#)

Es wurden Berechtigungen für Aktionen zur Observability-Verwaltung hinzugefügt, sodass nur Lesezugriff [CloudWatchLogsReadOnlyAccess](#) auf Telemetrie-Pipelines und S3-Tabellenintegrationen möglich ist.

2. Dezember 2025

[Aktualisierte CloudWatchLogsFullAccess Richtlinie](#)

Es wurden Berechtigungen für Aktionen zur Observability-Verwaltung hinzugefügt, sodass nur Lesezugriff [CloudWatchLogsFullAccess](#) auf Telemetrie-Pipelines und S3-Tabellenintegrationen möglich ist.

2. Dezember 2025

[Neue Funktion zur Protokollverwaltung in CloudWatch Logs](#)

CloudWatch konsolidiert die Protokollverwaltung in einem einzigen Dienst mit integrierten Governance-Funktionen, ohne dass mehrere Kopien derselben Daten in verschiedenen Tools und Datenspeichern gespeichert und verwaltet werden müssen. Weitere Informationen finden Sie unter [Protokollverwaltung](#).

2. Dezember 2025

[Neue Funktion zur Datenaufnahme und Normalisierung in Logs CloudWatch](#)

CloudWatch sammelt durch die Integration mit AWS Organizations und vorgefertigte Konnektoren für Quellen von Drittanbietern automatisch AWS verkaufte Logs für Konten und AWS Regionen. Weitere Informationen finden Sie unter [Transform logs during ingestion](#).

2. Dezember 2025

[Neue Funktionen für Analysen und Abfragen auf der Grundlage von Datenquellen in CloudWatch Logs Insights](#)

CloudWatch Insights führt Facetten für Abfragen ein. Facetten sind nützlich für die Analyse von Protokollen, da sie es Ihnen ermöglichen, Ihre Daten interaktiv zu filtern und zu gruppieren, ohne Abfragen ausführen zu müssen. Eine Facette ist ein Feld in Ihren Protokollen (wie ServiceName oder StatusCode), das das Filtern, Aggregieren und Analysieren über Protokollgruppen hinweg ermöglicht. Weitere Informationen finden Sie unter [Verwenden von Facetten, um Logs zu gruppieren und zu untersuchen](#).

2. Dezember 2025

[Neue Integration von S3-Tabellen](#)

Die S3-Tabellen-Integration mit CloudWatch ermöglicht Ihnen den Zugriff auf Protokolldaten, die CloudWatch mithilfe von Analyse-Engines wie Amazon Athena, Amazon Redshift und Tools von Drittanbietern aufgenommen wurden, die die Verbindung zu Apache Iceberg-compatible Stores unterstützen. Weitere Informationen finden Sie unter [Zugreifen auf](#) Protokolle mit der Integration von S3-Tabellen.

2. Dezember 2025

[CloudWatch Logs hat Unterstützung für die Cross-account regionsübergreifen de Protokollzentralisierung hinzugefügt](#)

CloudWatch Logs hat Unterstützung für die Cross-account regionsübergreifen de Protokollzentralisierung hinzugefügt, die es CloudWatch Logs ermöglicht, Protokolldaten von AWS Organizations Mitgliedskonten in einem zentralen Konto zur Analyse zu sammeln. Weitere Informationen finden Sie unter [Cross-accountRegionsübergreifende Protokollzentralisierung](#).

17. September 2025

[CloudWatch Protokolliert die Erkennung von Anomalien, aktualisierte Beispielrichtlinie](#)

CloudWatch Logs hat die KMS-Beispielrichtlinie aktualisiert, um ihren Geltungsbereich zu reduzieren und die Sicherheit zu verbessern, indem Bedingungen für `aws:SourceAccount` und `aws:SourceArn` hinzugefügt wurden. Weitere Informationen finden Sie unter [Verschlüsseln eines Anomaliedetektors und seiner Ergebnisse](#) mit KMS AWS

14. Juli 2025

[CloudWatch Logs Insights
bietet Unterstützung für
Amazon VPC Route Server-Pr
otokolle](#)

CloudWatch Logs Insights bietet Unterstützung für Amazon VPC Route Server-Protokolle. Weitere Informationen finden Sie unter [Aktivieren der Protokollierung von AWS Diensten](#). Die neue Protokollquelle für Amazon VPC Route Server EVENT_LOGS ist dokumentiert in [PutDelive
rySource](#)

12. Juni 2025

[CloudWatch Logs Insights
bietet Unterstützung für AWS
PCS Protokolle](#)

CloudWatch Logs Insights fügt Unterstützung für AWS PCS Protokolle hinzu. Weitere Informationen finden Sie unter [Aktivieren der Protokollierung von AWS Diensten](#). Die neuen Protokollquellen für AWS PCSPCS_SCHEDULER_LOGS und PCS_JOBCOMP_LOGS sind dokumentiert in [PutDelive
rySource](#)

12. Juni 2025

[CloudWatch Logs Insights fügt
Unterstützung für AWS Entity
Resolution Protokolle hinzu](#)

CloudWatch Logs Insights fügt Unterstützung für AWS Entity Resolution Protokolle hinzu. Weitere Informationen finden Sie unter [Aktivieren der Protokollierung von AWS Diensten](#). Die neue Protokollquelle für AWS Entity Resolution, WORKFLOW_LOGS ist dokumentiert in [PutDelive
rySource](#)

22. Mai 2025

[CloudWatch Protokolliert verwaltete Richtlinienaktualisierungen, um Zusammenfassungen in natürlicher Sprache zu unterstützen](#)

Berechtigungen für `cloudwatch:GenerateQueryResultsSummary` wurden hinzugefügt. `CloudWatchLogsFullAccess` und `CloudWatchLogsReadOnlyAccess` ermöglichen die Generierung einer Zusammenfassung der Abfrageergebnisse in natürlicher Sprache. Den Inhalt der Richtlinien finden Sie unter [CloudWatchLogsFullAccess](#) und [CloudWatchLogsReadOnlyAccess](#) im Referenzhandbuch für AWS verwaltete Richtlinien.

20. Mai 2025

[CloudWatch Logs Insights bietet Unterstützung für die Generierung von Zusammenfassungen in natürlicher Sprache aus CloudWatch Logs Insights-Abfrageergebnissen](#)

Unterstützung für Zusammenfassungen in natürlicher Sprache in CloudWatch Logs Insights wurde hinzugefügt. Diese Funktion generiert menschenlesbare Zusammenfassungen der Abfrageergebnisse, die derzeit in der Region USA Ost (Nord-Virginia) verfügbar sind. Weitere Informationen finden Sie unter [Generieren von Zusammenfassungen in natürlicher Sprache aus CloudWatch Logs Insights-Abfrageergebnissen](#).

20. Mai 2025

[CloudWatchOpenSearchDashboardsFullAccess](#) und [CloudWatchOpenSearchDashboardsFullAccess](#) sind neue IAM-Richtlinien

CloudWatch Logs fügte zwei neue IAM-Richtlinien hinzu, und CloudWatchOpenSearchDashboardsFullAccess. CloudWatchOpenSearchDashboardsAccess und CloudWatchOpenSearchDashboardsFullAccesserteilt die Erlaubnis, Integrationen mit OpenSearch Service zu erstellen und zu verwalten. CloudWatchOpenSearchDashboardsAccessgewährt Zugriff auf die Dashboards für verkaufte Logs, die in diesen Integrationen erstellt wurden. Weitere Informationen finden Sie unter [Vended-Log-Dashboards bereitgestellt von Amazon OpenSearch Service](#).

01. Dezember 2024

[CloudWatchLogsFullAccess](#) Richtlinie aktualisiert

CloudWatch Logs hat der CloudWatchLogsFullAccessRichtlinie Berechtigungen für Amazon OpenSearch Service und IAM hinzugefügt, um die Integration von CloudWatch Logs in den OpenSearch Service für einige Funktionen zu ermöglichen.

01. Dezember 2024

[CloudWatch Logs Insights
fügt der Abfragesyntax neue
Strukturtypen hinzu](#)

CloudWatch Logs Insights fügt den `unnest` Befehl und zwei JSON-Funktionen hinzu, mit denen Sie JSON-Zeichenketten als Maps und Listen verwenden können. Weitere Informationen finden Sie unter [Strukturtypen](#).

21. November 2024

[CloudWatch Logs unterstützt die Protokolltransformation während der Protokollaufnahme](#)

Sie können Protokolltransformatoren erstellen, mit denen Protokollereignisse zum Zeitpunkt der Aufnahme geändert werden können. Dies hilft Ihnen dabei, Ihre Protokolle in verschiedenen Formaten und aus verschiedenen Quellen in konsistente und kontextreiche Formate zu normalisieren. Weitere Informationen finden Sie unter [Transformieren](#) von Protokollen während der Aufnahme.

20. November 2024

[CloudWatch Logs Insights fügt die Feldindizierung hinzu](#)

CloudWatch Logs Insights hat Unterstützung für die Feldindizierung von Protokollen hinzugefügt. Wenn Sie dann einen Feldindex in einer CloudWatch Logs Insights-Abfrage verwenden, versucht die Abfrage, die Verarbeitung von Protokollereignissen zu überspringen, von denen bekannt ist, dass sie das indizierte Feld nicht enthalten. Weitere Informationen finden [Sie unter Feldindizes erstellen, um die Abfrageleistung zu verbessern und das Scanvolumen zu reduzieren.](#)

20. November 2024

[CloudWatch Logs Insights-Unterstützung für die Generierung von Abfragen in natürlicher Sprache ist allgemein verfügbar](#)

CloudWatch Logs Insights unterstützt natürliche Sprache zur Generierung und Aktualisierung von Abfragen. Weitere Informationen finden Sie unter [Verwenden natürlicher Sprache zum Generieren und Aktualisieren von CloudWatch Logs Insights-Abfragen.](#)

20. Juni 2024

[CloudWatchLogsReadOnlyAccessRichtlinie aktualisiert](#)

CloudWatch Logs hat die `cloudwatch:GenerateQuery` Berechtigung zu hinzugefügt `CloudWatchLogsReadOnlyAccess`, sodass Benutzer mit dieser Richtlinie anhand einer Aufforderung in natürlicher Sprache eine [CloudWatch Logs Insights-Abfragezeichenfolge](#) generieren können.

26. November 2023

[CloudWatchLogsFullAccessDie Richtlinie wurde aktualisiert](#)

CloudWatch Logs hat die `cloudwatch:GenerateQuery` Berechtigung zu hinzugefügt `CloudWatchLogsFullAccess`, sodass Benutzer mit dieser Richtlinie anhand einer Aufforderung in natürlicher Sprache eine [CloudWatch Logs Insights-Abfragezeichenfolge](#) generieren können.

26. November 2023

[CloudWatch Logs fügt eine Protokollmusteranalyse hinzu](#)

CloudWatch Logs sucht jetzt jedes Mal, wenn Sie eine CloudWatch Logs Insights-Abfrage ausführen, nach Mustern in Protokollereignissen. Weitere Informationen finden Sie unter [Musteranalyse](#).

26. November 2023

[CloudWatch Logs fügt die Erkennung von Protokollanomalien hinzu](#)

Sie können einen Detektor für Protokollanomalien für eine Protokollgruppe erstellen. Der Anomaliedetektor scannt die in die Protokollgruppe aufgenommenen Protokollereignisse und findet Anomalien in den Protokolldaten. Unter [Protokollanomalieerkennung](#) finden Sie weitere Informationen.

26. November 2023

[CloudWatch Logs fügt eine Vergleichsfunktion hinzu](#)

Sie können CloudWatch Logs Insights jetzt verwenden, um Änderungen Ihrer Protokollereignisse im Laufe der Zeit zu vergleichen. Weitere Informationen finden Sie unter [Vergleich \(Diff\) mit früheren Zeiträumen](#).

26. November 2023

[CloudWatch Logs fügt eine neue Protokollklasse hinzu](#)

CloudWatch Logs unterstützt zwei Klassen von Protokollgruppen, sodass Sie eine kostengünstige Option für Protokolle haben, auf die Sie selten zugreifen, und Sie haben auch eine Option mit vollem Funktionsumfang für Protokolle, die eine Echtzeitüberwachung oder andere Funktionen erfordern. Weitere Informationen finden Sie unter [Protokollklassen](#).

26. November 2023

[CloudWatch Logs Insights unterstützt die Generierung von Abfragen in natürlicher Sprache](#)

CloudWatch Logs Insights unterstützt natürliche Sprache zur Generierung und Aktualisierung von Abfragen. Weitere Informationen finden Sie unter [Verwenden natürlicher Sprache zum Generieren und Aktualisieren von CloudWatch Logs Insights-Abfragen](#).

26. November 2023

[CloudWatch Logs fügt Unterstützung für die Syntax von Filtermustern für reguläre Ausdrücke für Live Tail hinzu](#)

Mit flexiblen regulären Ausdrücken innerhalb von Live-Tail-Filtermustern können Sie Ihre Such- und Abgleichsvorgänge jetzt weiter an Ihre Bedürfnisse anpassen. Weitere Informationen finden Sie unter [Syntax von Filtermustern](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

13. November 2023

[CloudWatch Logs fügt Unterstützung für die Syntax von Filtermustern mit regulären Ausdrücken für Metrikfilter, Abonnementfilter und Filter-Protokollereignisse hinzu](#)

Mit flexiblen regulären Ausdrücken innerhalb von Filtermustern können Sie Ihre Such- und Abgleichsvorgänge jetzt weiter an Ihre Bedürfnisse anpassen. Weitere Informationen finden Sie unter [Syntax von Filtermustern](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

5. September 2023

[CloudWatch Logs Insights fügt einen Musterbefehl hinzu](#)

Sie können jetzt Muster in Ihren CloudWatch Logs Insights-Abfragen verwenden , um Ihre Protokolldaten automatisch zu Mustern zu gruppieren. Ein Muster ist eine gemeinsam genutzte Textstruktur, die sich in Ihren Protokollfeldern wiederholt. Weitere Informationen finden Sie unter [Muster](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

17. Juli 2023

[CloudWatch Logs Insights fügt einen Dedup-Befehl hinzu](#)

Sie können jetzt Dedup in Ihren CloudWatch Logs Insights-Abfragen verwenden , um doppelte Ergebnisse zu entfernen, die auf bestimmten Werten in von Ihnen angegebenen Feldern basieren. Weitere Informationen finden Sie unter [Dedup](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

20. Juni 2023

[Account-level Datenschutzzrichtlinien](#)

Sie können jetzt Datenschutzzrichtlinien auf Kontoebene festlegen. Mit diesen Richtlinien auf Kontoebene können vertrauliche Informationen in Protokollereignissen in allen Protokollgruppen des Kontos geprüft und maskiert werden. Weitere Informationen finden Sie unter [Hilfe zum Schutz sensibler Protokolldaten durch Maskierung](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

08. Juni 2023

[Live-Tail-Feature hinzugefügt](#)

CloudWatch Für Protokolle wurde die Live-Tail-Funktion hinzugefügt, sodass Sie Protokolle scannen können, während sie aufgenommen werden, um bei der Fehlerbehebung zu helfen. Sie können den angezeigten Stream von Protokollereignissen optional nach bestimmten Begriffen filtern und auch Protokollereignisse hervorheben, die bestimmte Begriffe enthalten. Weitere Informationen finden Sie unter [Use live tail to view logs in near real time](#).

6. Juni 2023

[CloudWatchLogsRead
OnlyAccessRichtlinie aktualisiert](#)

CloudWatch Protokolliert hinzugefügte Berechtigungen für CloudWatchLogsRead OnlyAccess. Die logs:StopLiveTail Berechtigungen logs:StartLiveTail und wurden hinzugefügt, sodass Benutzer mit dieser Richtlinie die Konsole verwenden können, um CloudWatch Logs-Live-Tail-Sitzungen zu starten und zu beenden. Weitere Informationen finden Sie unter [Use live tail to view logs in near real time](#).

6. Juni 2023

[CloudWatch Logs Insights veröffentlicht](#)

Sie können CloudWatch Logs Insights verwenden, um Ihre Protokolldaten interaktiv zu suchen und zu analysieren. Weitere Informationen finden Sie unter [Analysieren von Protokolldaten mit CloudWatch Logs Insights](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

27. November 2018

Unterstützung für Amazon VPC-Endpunkte

Sie können jetzt eine private Verbindung zwischen Ihrer VPC und CloudWatch Logs herstellen. Weitere Informationen finden Sie unter [Using CloudWatch Logs with Interface VPC Endpoints](#) im Amazon CloudWatch Logs-Benutzerhandbuch.

28. Juni 2018

In der folgenden Tabelle werden die wichtigen Änderungen am Amazon CloudWatch Logs-Benutzerhandbuch beschrieben.

Änderungen	Beschreibung	Datum der Veröffentlichung
Schnittstellen-VPC-Endpunkte	In einigen Regionen können Sie einen VPC-Schnittstellen-Endpunkt verwenden, um zu verhindern, dass der Datenverkehr zwischen Ihrer Amazon VPC und CloudWatch Logs das Amazon-Netzwerk verlässt. Weitere Informationen finden Sie unter Verwenden von CloudWatch Protokollen mit VPC-Endpunkten der Schnittstelle .	7. März 2018
Route-53-DNS-Abfrageprotokolle	Sie können CloudWatch Logs verwenden, um Protokolle über die DNS-Abfragen zu speichern, die von Route 53 empfangen wurden. Weitere Informationen finden Sie unter Was ist Amazon CloudWatch Logs? oder Protokollieren von DNS-Abfragen im Entwicklerhandbuch für Amazon Route 53.	7. September 2017
Markieren von Protokollgruppen	Sie können Tags verwenden, um Ihre Protokollgruppen zu kategorisieren. Weitere Informationen finden Sie unter Taggen Sie Protokollgruppen in Amazon CloudWatch Logs .	13. Dezember 2016

Änderungen	Beschreibung	Datum der Veröffentlichung
Verbesserungen an der Konsole	Sie können von Metrikdiagrammen zu den zugehörigen Protokollgruppen navigieren. Weitere Informationen finden Sie unter Wechseln von Metriken zu Protokollen .	7. November 2016
Verbesserungen an der Benutzerfreundlichkeit der Konsole	Verbesserte Benutzererfahrung für einfacheres Suchen, Filtern und Fehlerbeheben. Beispielsweise können Sie jetzt Ihre Protokolldaten nach Datum und Uhrzeit filtern. Weitere Informationen finden Sie unter An Logs gesendete Protokolldaten anzeigen CloudWatch .	29. August 2016
AWS CloudTrail Unterstützung für Amazon CloudWatch Logs und neue CloudWatch Logs-Metriken hinzugefügt	AWS CloudTrail Unterstützung für CloudWatch Logs hinzugefügt. Weitere Informationen finden Sie unter Logging CloudWatch protokolliert API- und Konsolenoperationen in AWS CloudTrail .	10. März 2016
Unterstützung für den Export von CloudWatch Logs nach Amazon S3 hinzugefügt	Unterstützung für den Export von CloudWatch Logs-Daten nach Amazon S3 hinzugefügt. Weitere Informationen finden Sie unter Exportieren von Protokolldaten nach Amazon S3 .	7. Dezember 2015

Änderungen	Beschreibung	Datum der Veröffentlichung
Unterstützung für AWS CloudTrail protokollierte Ereignisse in Amazon CloudWatch Logs hinzugefügt	Sie können Alarme in bestimmten API-Aktivitäten erstellen CloudWatch und Benachrichtigungen über bestimmte API-Aktivitäten erhalten, die von erfasst wurden, CloudTrail und die Benachrichtigung zur Fehlerbehebung verwenden.	10. November 2014
Unterstützung für Amazon CloudWatch Logs hinzugefügt	Sie können Amazon CloudWatch Logs verwenden , um Ihre System-, Anwendungs- und benutzerdefinierten Protokolldateien von Amazon Elastic Compute Cloud (Amazon EC2) -Instances oder anderen Quellen aus zu überwachen, zu speichern und darauf zuzugreifen. Anschließend können Sie die zugehörigen Protokolldaten mithilfe der CloudWatch Amazon-Konsole, der CloudWatch CloudWatch Logs-Befehle im oder des AWS CLI Logs-SDK aus CloudWatch Logs abrufen. Weitere Informationen finden Sie unter Was ist Amazon CloudWatch Logs? .	10. Juli 2014

AWS Glossar

Die neueste AWS Terminologie finden Sie im [AWS Glossar](#) in der AWS-Glossar Referenz.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.