aws

Leitfaden

CloudWatch Amazon-Protokolle

CloudWatch Amazon-Protokolle: Leitfaden

Table of Contents

Was ist Amazon CloudWatch Logs?

Sie können Amazon CloudWatch Logs verwenden, um Ihre Protokolldateien von Amazon Elastic Compute Cloud (Amazon EC2) -Instances, Route 53 und anderen Quellen zu überwachen AWS CloudTrail, 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 — Sie können CloudWatch Logs Insights verwenden, um Ihre Logdaten interaktiv zu suchen und zu analysieren. Sie können Abfragen durchführen, um effizienter und effektiver auf betriebliche Probleme reagieren zu können. CloudWatch Logs Insights enthält eine speziell entwickelte Abfragesprache mit einigen einfachen, aber leistungsstarken Befehlen. Wir stellen Beispielabfragen, Befehlsbeschreibungen, automatische Abfragevervollständigung und Protokollfelderkennung 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.

- 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 Verwenden von Live Tail zur Anzeige von Protokollen in nahezu Echtzeit.
- Protokolle von Amazon EC2 EC2-Instances überwachen — Sie können CloudWatch Logs verwenden, um Anwendungen und Systeme anhand von Protokolldaten zu überwachen. Mit CloudWatch Logs können Sie 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 "NullReferenceException„) 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, CloudWatch meldet 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. Informationen zu den ersten Schritten 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.

- Protokolldaten archivieren — Sie können CloudWatch Logs verwenden, um Ihre Protokolldaten auf einem äußerst langlebigen 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.

Verwandte AWS Dienste

Die folgenden Dienste werden in Verbindung mit CloudWatch Logs verwendet:

- AWS CloudTrailist ein Webdienst, mit dem Sie die Aufrufe der CloudWatch Logs-API für Ihr Konto überwachen können, einschließlich der Aufrufe von AWS Management Console, 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 unterProtokollieren von Amazon-CloudWatch-Logs-API-Aufrufen in AWS CloudTrail.
- AWS Identity and Access Management (IAM) ist ein Webdienst, 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 Protokollereignisdatensatz, den CloudWatch Logs versteht, enthält zwei Eigenschaften: den Zeitstempel, zu dem das Ereignis eingetreten ist, und die unformatierte Ereignisnachricht. 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 Log-Stream 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 metrische Beobachtungen aus aufgenommenen Ereignissen zu extrahieren und sie 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 können Sie angeben, wie lange Protokollereignisse in CloudWatch Protokollen 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.

Amazon CloudWatch Logs – Abrechnung und Kosten

Ausführliche Informationen zur Analyse Ihrer Kosten und Nutzung für CloudWatch Logs und CloudWatch sowie bewährte Methoden zur Reduzierung Ihrer Kosten finden Sie unter CloudWatch billing and cost.

Weitere Informationen zu Preisen finden Sie unter Amazon CloudWatch – Preise.

Wenn Sie sich bei AWS registrieren, können Sie kostenlos mit der Verwendung von CloudWatch Logs beginnen, indem Sie das kostenlose Kontingent von AWS nutzen.

Die Standard-Preise gelten für Protokolle, die von anderen Services unter Verwendung von CloudWatch Logs gespeichert werden (z. B. Amazon-VPC-Flow-Protokolle und Lambda-Protokolle).

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.

	Standard	Seltener Zugriff	
Vollständig verwaltete Protokollaufnahme und Speicherung	✓	✓	
Kontoübergreifende Funktionen	✓	✓	
Verschlüsselung mit AWS KMS	✓	✓	
CloudWatch Logs Insights-Abfragebefehle	✓	✓ (Die meisten Befehle — siehe Befehle, die in Protokollklassen unterstützt werden.)	
CloudWatch Logs Insights entdeckte Felder	✓		
Unterstützung bei Abfragen in natürlicher Sprache	✓		
CloudWatch Protokolliert die Erkennung von Anomalien	✓		
Mit dem vorherigen Zeitraum vergleichen	✓		
Abonnementfilter	✓		
Exportieren zu Amazon S3	✓		
GetLogEvents und FilterLogEvents API-Operationen	✓	Nicht unterstützt Verwenden Sie CloudWatch	

	Standard	Seltener Zugriff	
		Logs Insights, um Protokoll ereignisse anzuzeige n, die in Protokoll gruppen der Protokoll klasse für seltenen Zugriff gespeichert sind.	
Metrische Filter	✓		
Erfassung von Container-Insights-Protokollen	✓		
Erfassung von Lambda Insights-Protokollen	✓		
Schutz sensibler Daten mit Maskierung	✓		
Format für eingebettete Metriken	✓		

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

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

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Aus Sicherheitsgründen sollten Sie einem Benutzer Administratorzugriff zuweisen und nur den Root-Benutzer verwenden, um Aufgaben auszuführen, für die Root-Benutzerzugriff erforderlich ist.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen Sie einen Benutzer mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen Sie einen Benutzer mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.

2. Gewähren Sie einem Benutzer in IAM Identity Center Administratorzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit der Standardeinstellung konfigurieren IAM-Identity-Center-Verzeichnis im AWS IAM Identity Center Benutzerhandbuch.

Melden Sie sich als Benutzer mit Administratorzugriff an

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Zugriffsportal.

Weisen Sie weiteren Benutzern Zugriff zu

1. Erstellen Sie in IAM Identity Center einen Berechtigungssatz, der der bewährten Methode zur Anwendung von Berechtigungen mit den geringsten Rechten folgt.

   Anweisungen finden Sie im Benutzerhandbuch unter Einen Berechtigungssatz erstellen.AWS IAM Identity Center

2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Anweisungen finden Sie im AWS IAM Identity Center Benutzerhandbuch unter Gruppen hinzufügen.

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

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 Instance Metadata Service Version 2 (IMDSv2) nicht. Wenn Ihr Server IMDSv2 verwendet, müssen Sie den neueren Unified Agent anstelle des älteren Logs-Agenten verwenden. CloudWatch
Der Rest dieses Abschnitts erklärt die Verwendung des älteren CloudWatch Logs-Agenten für Kunden, 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 des Instance Metadata Service (IMDSsv2) nicht. Wenn Ihr Server IMDSv2 verwendet, 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-Agent 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 die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Rollen aus.
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 Verbinden mit Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Linux-Instances.

   Weitere Informationen zu Verbindungsproblemen finden Sie unter Beheben von Verbindungsproblemen mit Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Linux-Instances.

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 Referenz zum CloudWatch-Logs-Agenten.

5. Standardmäßig verweist `/etc/awslogs/awscli.conf` auf die Region us-east-1. Um Ihre Protokolle in eine andere Region zu verschieben, bearbeiten Sie die Datei `awscli.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. 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 .

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 Verbinden mit Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Linux-Instances.

   Weitere Informationen zu Verbindungsproblemen finden Sie unter Beheben von Verbindungsproblemen mit Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Linux-Instances.
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	Beschreibung
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.
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.

Item	Beschreibung
Pfad der hochzulad enden Protokolldatei	Der Speicherort der Datei, der die Protokolldaten enthält, die Sie senden möchten. Das Installationsprogramm schlägt Ihnen einen Pfad vor.
Ziel-Protokollgrup penname	Der Name für Ihre Protokollgruppe. Das Installationsprogramm schlägt Ihnen einen Protokollgruppenamen vor.
Ziel-Protokoll-Str eamname	Standardmäßig ist dies der Name des Hosts. Das Installationsprogr amm schlägt Ihnen einen Hostnamen vor.
Zeitstempelformat	Geben Sie das Format des Zeitstempels innerhalb der angegeben en 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 Referenz zum CloudWatch-Logs-Agenten.

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

3. 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: 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 Referenz zum CloudWatch-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 die IAM-Konsole unter https://console.aws.amazon.com/iam/.
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 zu benutzerdefinierten Richtlinien finden Sie unter IAM-Richtlinien für Amazon EC2 im Amazon-EC2-Benutzerhandbuch für Linux-Instances.
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:

```
{
    "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:::myawsbucket/*"
            ]
        }
    ]
}
```

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 Create Role (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 unter Launchen einer Instance im Amazon-EC2-Benutzerhandbuch für Linux-Instances.

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 3Konfigurieren 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://DOC-EXAMPLE-BUCKET1/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, den neueren Unified CloudWatch Agent zu verwenden. 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 für jede ID eine andere Region 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 Protokollgruppen 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 Zeitzoneninformationen bereitstellen, wenn der Zeitstempel Ihres Protokolls keine Zeitzoneninformationen enthält. Wenn dieser Parameter leer gelassen wird und Ihr Zeitstempel keine Zeitzoneninformationen enthält, verwendet CloudWatch Logs standardmäßig die lokale Zeitzone. Dieser Parameter wird ignoriert, wenn der Zeitstempel bereits Zeitzoneninformationen 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\W3SVC`*n*).

> **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 Zeitzoneninformationen bereitstellen, wenn der Zeitstempel Ihres Protokolls keine Zeitzoneninformationen enthält. Wenn dieser Parameter leer gelassen wird und Ihr Zeitstempel keine Zeitzoneninformationen enthält, verwendet CloudWatch Logs standardmäßig die lokale Zeitzone. Dieser Parameter wird ignoriert, wenn der Zeitstempel bereits Zeitzoneninformationen 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 Zuordnen dieser Rolle zu einer vorhandenen Instance finden Sie unter Anfügen einer IAM-Rolle an eine Instance im Amazon-EC2-Benutzerhandbuch für Windows-Instances.

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 als Befehlsdokument AWS- ausConfigureCloudWatch.
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-Benutzerhandbuch für Windows-Instances.
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, den neueren Unified CloudWatch Agent zu verwenden. 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 für jede ID eine andere Region 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 Protokollgruppen 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 Zeitzoneninformationen bereitstellen, wenn der Zeitstempel Ihres Protokolls keine Zeitzoneninformationen enthält. Wenn dieser Parameter leer gelassen wird und Ihr Zeitstempel keine Zeitzoneninformationen enthält, verwendet CloudWatch Logs standardmäßig die lokale Zeitzone. Dieser Parameter wird ignoriert, wenn der Zeitstempel bereits Zeitzoneninformationen 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\W3SVC`*n*).

> **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 Zeitzoneninformationen bereitstellen, wenn der Zeitstempel Ihres Protokolls keine Zeitzoneninformationen enthält. Wenn dieser Parameter leer gelassen wird und Ihr Zeitstempel keine Zeitzoneninformationen enthält, verwendet CloudWatch Logs standardmäßig die lokale Zeitzone. Dieser Parameter wird ignoriert, wenn der Zeitstempel bereits Zeitzoneninformationen 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 Logs zu CloudWatch ermöglichen, verwenden Sie den Dienst EC2Config (. `EC2Config.exe`) Wenn Ihre Instance EC2Config-4.0 oder höher aufweist, können Sie diese Anleitung verwenden. Weitere Informationen zur Verwendung einer früheren Version von EC2Config finden Sie unter Verwenden von EC2Config 3.x oder früher zur Konfiguration CloudWatch im Amazon EC2 EC2-Benutzerhandbuch für Windows-Instances

So konfigurieren Sie mit EC2Config 4.x CloudWatch

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 Befehl neu: PowerShell

```
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.

Schnellstart: Installieren Sie den CloudWatch Logs-Agenten mithilfe von AWS OpsWorks and Chef

Sie können den CloudWatch Logs-Agent installieren und Log-Streams mithilfe von AWS OpsWorks Chef, einem Automatisierungstool für Systeme und Cloud-Infrastrukturen von Drittanbietern, erstellen. Chef verwendet sog. „Rezepte", die Sie schreiben, um Software auf Ihrem Computer zu installieren und zu konfigurieren, und sog. „Rezeptbücher", also Rezeptsammlungen, mit denen die Konfigurations- und Richtlinienverteilungsaufgaben des Tools ausgeführt werden. Weitere Informationen finden Sie unter Chef.

Die nachstehenden Beispiele für Chef-Rezepte zeigen, wie Sie eine Protokolldatei auf jeder EC2-Instance überwachen. Für die Rezepte wird der Stacknamen als Protokollgruppe und der Hostnamen der Instance als Protokollstreamnamen verwendet. Um mehrere Protokolldateien zu überwachen, müssen Sie die Rezepte erweitern, um mehrere Protokollgruppen und Protokollstreams zu erstellen.

Schritt 1: Erstellen von benutzerdefinierten Rezepten

Erstellen Sie ein Repository zum Speichern Ihrer Rezepte. AWS OpsWorks unterstützt Git und Subversion, oder Sie können ein Archiv in Amazon S3 speichern. Die Struktur des Rezeptbuch-Repository wird unter Rezeptbuch-Repositorys im AWS OpsWorks -Benutzerhandbuch beschrieben. Bei den nachstehenden Beispielen wird davon ausgegangen, dass das Rezeptbuch `logs` heißt. Das Rezept install.rb installiert den CloudWatch Logs-Agenten. Sie können auch das Kochbuch-Beispiel (CloudWatchLogs-Cookbooks.zip) herunterladen.

Erstellen Sie eine Datei mit dem Namen metadata.rb, die folgenden Code enthält:

```
#metadata.rb
```

```
name             'logs'
version          '0.0.1'
```

Erstellen Sie die CloudWatch Logs-Konfigurationsdatei:

```
#config.rb

template "/tmp/cwlogs.cfg" do
  cookbook "logs"
  source "cwlogs.cfg.erb"
  owner "root"
  group "root"
  mode 0644
end
```

Laden Sie den CloudWatch Logs-Agenten herunter und installieren Sie ihn:

```
# install.rb

directory "/opt/aws/cloudwatch" do
  recursive true
end

remote_file "/opt/aws/cloudwatch/awslogs-agent-setup.py" do
  source "https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-
setup.py"
  mode "0755"
end

  execute "Install CloudWatch Logs agent" do
  command "/opt/aws/cloudwatch/awslogs-agent-setup.py -n -r region -c /tmp/cwlogs.cfg"
  not_if { system "pgrep -f aws-logs-agent-setup" }
end
```

> **Note**
>
> Ersetzen Sie im obigen Beispiel *region* mit einem der folgenden: 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.

Wenn die Installation des Agenten fehlschlägt, prüfen Sie, ob das Paket `python-dev` installiert ist. Wenn dies nicht der Fall ist, verwenden Sie den folgenden Befehl und wiederholen Sie die Agenteninstallation:

```
sudo apt-get -y install python-dev
```

Dieses Rezept verwendet eine Vorlagendatei namens cwlogs.cfg.erb, die Sie ändern können, um verschiedene Attribute festzulegen, wie z. B. welche Dateien protokolliert werden sollen. Weitere Informationen zu diesen Attributen finden Sie unter Referenz zum CloudWatch-Logs-Agenten.

```
[general]
# Path to the AWSLogs agent's state file. Agent uses this file to maintain
# client side state across its executions.
state_file = /var/awslogs/state/agent-state


## Each log file is defined in its own section. The section name doesn't
## matter as long as its unique within this file.
#
#[kern.log]
#
## Path of log file for the agent to monitor and upload.
#
#file = /var/log/kern.log
#
## Name of the destination log group.
#
#log_group_name = kern.log
#
## Name of the destination log stream.
#
#log_stream_name = {instance_id}
#
## Format specifier for timestamp parsing.
#
#datetime_format = %b %d %H:%M:%S
#
#

[<%= node[:opsworks][:stack][:name] %>]
datetime_format = [%Y-%m-%d %H:%M:%S]
log_group_name = <%= node[:opsworks][:stack][:name].gsub(' ','_') %>
```

```
file = <%= node[:cwlogs][:logfile] %>
log_stream_name = <%= node[:opsworks][:instance][:hostname] %>
```

Die Vorlage erhält den Stack-Namen und den Host-Namen, indem auf die entsprechenden Attribute in der Stack-Konfiguration und Bereitstellungs-JSON verwiesen wird. Das Attribut, das die Datei zu protokollierende Datei festlegt, ist in der Attributdatei des cwlogs-Rezeptbuchs „default.rb" definiert (logs/attributes/default.rb).

```
default[:cwlogs][:logfile] = '/var/log/aws/opsworks/opsworks-agent.statistics.log'
```

Schritt 2: Erstellen Sie einen AWS OpsWorks Stack

1. Öffnen Sie die AWS OpsWorks Konsole unter https://console.aws.amazon.com/opsworks/.
2. Wählen Sie im OpsWorks Dashboard Stapel hinzufügen aus, um einen AWS OpsWorks Stack zu erstellen.
3. Wählen Sie auf dem Bildschirm Add stack die Option Chef 11 stack aus.
4. Geben Sie für Stack name einen Namen ein.
5. Wählen Sie bei Use custom Chef Cookbooks die Option Yes.
6. Wählen Sie für Repository type den Repository-Typ, den Sie verwenden. Wenn Sie das oben genannte Beispiel verwenden, wählen Sie Http Archive.
7. Geben Sie für Repository URL das Repository ein, in dem Sie das Rezeptbuch gespeichert haben, das Sie im vorherigen Schritt erstellt haben. Wenn Sie das oben genannte Beispiel verwenden, geben Sie **`https://s3.amazonaws.com/aws-cloudwatch/downloads/CloudWatchLogs-Cookbooks.zip`** ein.
8. Wählen Sie Add Stack aus, um den Stack zu erstellen.

Schritt 3: Erweitern Sie Ihre IAM-Rolle

Um CloudWatch Logs mit Ihren AWS OpsWorks Instances verwenden zu können, müssen Sie die von Ihren Instances verwendete IAM-Rolle erweitern.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Policies und Create Policy aus.
3. Wählen Sie auf der Seite Create Policy unter Create Your Own Policy die Option Select aus. Weitere Informationen zu benutzerdefinierten Richtlinien finden Sie unter IAM-Richtlinien für Amazon EC2 im Amazon-EC2-Benutzerhandbuch für Linux-Instances.

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:

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

6. Wählen Sie Richtlinie erstellen aus.
7. Wählen Sie im Navigationsbereich Rollen und dann im Inhaltsbereich für Rollenname den Namen der Instance-Rolle aus, die von Ihrem AWS OpsWorks Stack verwendet wird. Sie finden den von Ihrem Stack verwendeten Namen in den Stack-Einstellungen (der Standardnamen lautet `aws-opsworks-ec2-role`).

   > **Note**
   >
   > Wählen Sie den Rollennamen, aktivieren Sie nicht das Kontrollkästchen.

8. Wählen Sie auf der Registerkarte Permissions unter Managed Policies die Option Attach Policy aus.
9. Wählen Sie auf der Seite Attach Policy im Tabellen-Header (neben Filter und Search) die Option Policy Type und Customer Managed Policies aus.
10. Wählen Sie für Customer Managed Policies (Kundenverwaltete Richtlinien) die IAM-Richtlinie, die Sie zuvor erstellt haben, und wählen Sie Attach Policy (Richtlinie hinzufügen) aus.

    Weitere Informationen zu Benutzern und Richtlinien finden Sie unter IAM-Benutzer und -Gruppen sowie unter Verwalten von IAM-Richtlinien im IAM-Benutzerhandbuch.

Schritt 4: Hinzufügen einer Ebene

1. Öffnen Sie die AWS OpsWorks Konsole unter https://console.aws.amazon.com/opsworks/.
2. Wählen Sie im Navigationsbereich Ebenen aus.
3. Wählen Sie im Inhaltsbereich einen Layer aus und dann Add layer.
4. Wählen Sie auf der OpsWorksRegisterkarte für Layer-Typ die Option Benutzerdefiniert aus.
5. Geben Sie für die Felder Name und Short name den Lang- und Kurznamen des Layers ein und wählen Sie dann Add layer aus.
6. Auf der Registerkarte Rezepte gibt es unter Custom Chef Recipes mehrere Überschriften — Setup, Configure, Deploy, Undeploy und Shutdown —, die den Ereignissen im Lebenszyklus entsprechen AWS OpsWorks . AWS OpsWorks löst diese Ereignisse an diesen wichtigen Punkten im Lebenszyklus der Instanz aus, wodurch die zugehörigen Rezepte ausgeführt werden.

   > **Note**
   >
   > Wenn die oben genannten Überschriften nicht sichtbar sind, wählen Sie unter Custom Chef Recipes die Option edit.

7. Geben Sie neben Setup den Text logs::config, logs::install ein und wählen Sie +, um ihn der Liste hinzuzufügen, wählen Sie anschließend Save.

   AWS OpsWorks führt dieses Rezept auf jeder der neuen Instanzen in dieser Ebene aus, direkt nachdem die Instanz gestartet wurde.

Schritt 5: Hinzufügen einer Instance

Mit dem Layer wird nur gesteuert, wie Instances konfiguriert werden. Sie müssen nun dem Layer einige Instances hinzufügen und diese starten.

1. Öffnen Sie die AWS OpsWorks Konsole unter https://console.aws.amazon.com/opsworks/.
2. Wählen Sie im Navigationsbereich Instances aus und dann unter Ihrem Layer die Option + Instance.
3. Akzeptieren Sie die Standardeinstellungen und wählen Sie Add Instance, um dem Layer die Instance hinzuzufügen.
4. Klicken Sie in der Spalte Actions auf start, um die Instance zu starten.

AWS OpsWorks startet eine neue EC2-Instance und konfiguriert Logs CloudWatch . Der Status der Instance ändert sich zu „online”, sobald sie bereit ist.

Schritt 6: Anzeigen von Protokollen

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 .

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 Verbinden mit Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Linux-Instances.

   Weitere Informationen zu Verbindungsproblemen finden Sie unter Beheben von Verbindungsproblemen mit Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Linux-Instances.
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 Verbinden mit Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Linux-Instances.

   Weitere Informationen zu Verbindungsproblemen finden Sie unter Beheben von Verbindungsproblemen mit Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Linux-Instances.
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-Agent

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 Verbinden mit Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Linux-Instances.

   Weitere Informationen zu Verbindungsproblemen finden Sie unter Beheben von Verbindungsproblemen mit Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Linux-Instances.
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
```

Schnellstart: Verwenden Sie diese Option AWS 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 AWS 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. AWS 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.

CloudWatch Logs mit einem AWS SDK verwenden

AWS Software Development Kits (SDKs) sind für viele gängige 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 for C++	AWS SDK for C++ Codebeispiele
AWS CLI	AWS CLI Codebeispiele
AWS SDK for Go	AWS SDK for Go Codebeispiele
AWS SDK for Java	AWS SDK for Java Codebeispiele
AWS SDK for JavaScript	AWS SDK for JavaScript Codebeispiele
AWS SDK for Kotlin	AWS SDK for Kotlin Codebeispiele
AWS SDK for .NET	AWS SDK for .NET Codebeispiele
AWS SDK for PHP	AWS SDK for PHP Codebeispiele
AWS Tools for PowerShell	Tools für PowerShell Codebeispiele
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) Codebeispiele
AWS SDK for Ruby	AWS SDK for Ruby Codebeispiele
AWS SDK for Rust	AWS SDK for Rust Codebeispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Codebeispiele
AWS SDK for Swift	AWS SDK for Swift Codebeispiele

Spezifische Beispiele für CloudWatch Logs finden Sie unterCodebeispiele 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.

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, die Ihnen helfen, schnell und effektiv auf betriebliche Probleme zu reagieren. Wenn ein Problem auftritt, können Sie CloudWatch Logs Insights verwenden, um mögliche Ursachen zu identifizieren und bereitgestellte Fixes zu validieren.

CloudWatch Logs Insights enthält eine speziell entwickelte Abfragesprache mit einigen einfachen, aber leistungsstarken Befehlen. CloudWatch Logs Insights bietet Beispielabfragen, Befehlsbeschreibungen, automatische Vervollständigung von Abfragen und Erkennung von Protokollfeldern, um Ihnen den Einstieg zu erleichtern. Beispielabfragen sind für verschiedene Arten von AWS -Serviceprotokollen enthalten.

CloudWatch Logs Insights erkennt automatisch Felder 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.

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

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

Sie können 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 KI-gestützte Funktion generiert auf der Grundlage Ihrer Aufforderung eine Abfrage und line-by-line erklärt, wie die Abfrage funktioniert. Weitere Informationen finden Sie unter Verwenden natürlicher Sprache zum Generieren und Aktualisieren von CloudWatch Logs Insights-Abfragen.

Wenn Sie mit einem Konto angemeldet sind, das als Überwachungskonto für CloudWatch kontoübergreifende Observability eingerichtet ist, 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 CloudWatch kontoübergreifende Beobachtbarkeit.

Eine einzelne Anforderung kann bis zu 50 Protokollgruppen abfragen. Nicht abgeschlossene Abfragen werden nach 60 Minuten beendet. Abfrageergebnisse sind 7 Tage lang verfügbar.

Sie können Abfragen speichern, die Sie erstellt haben. Dies kann Ihnen helfen, bei Bedarf komplexe Abfragen auszuführen, ohne sie jedes Mal neu erstellen zu müssen, wenn Sie diese ausführen möchten.

CloudWatch Für Logs Insights-Abfragen fallen Gebühren an, die auf der abgefragten Datenmenge basieren. Weitere Informationen finden Sie unter CloudWatch Amazon-Preise.

> **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 StartQueryin der Amazon CloudWatch Logs API-Referenz.

Inhalt

Befehle, die in Protokollklassen unterstützt werden

Alle CloudWatch Logs Insights-Abfragebefehle werden für Protokollgruppen in der Standard-Protokollklasse unterstützt. Protokollgruppen der Protokollklasse für seltenen Zugriff unterstützen alle Abfragebefehle mit Ausnahme von `patterndiff`, und`unmask`.

Erste Schritte: Tutorials zu Abfragen

Die folgenden Abschnitte enthalten Beispiel-Tutorials für Abfragen, die Ihnen die ersten Schritte mit CloudWatch Logs Insights erleichtern sollen.

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 aus und sehen dann, wie Sie sie ändern und erneut ausführen können.

Um eine Abfrage ausführen zu können, müssen Sie bereits Logs in CloudWatch 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 Sie 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 unterErste 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, die die 20 letzten 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 unterKlassen 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 Logs Insights-Abfragen und führen Sie sie erneut CloudWatch 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 entdeckt 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. Wählen Sie Ausführen aus.

   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.

CloudWatch Logs Insights bietet leistungsstarke Abfragefunktionen, darunter mehrere Befehle und Unterstützung für reguläre Ausdrücke sowie mathematische und statistische Operationen. Weitere Informationen finden Sie unter CloudWatch Syntax der Logs Insights-Abfrage.

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. 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.
4. Löschen Sie die Standardabfrage im Abfrage-Editor und geben Sie den folgenden Befehl ein:

```
stats count(*) by fieldName
```

5. Ersetzen Sie *fieldName* mit einem erkannten Feld aus dem Menü Fields (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.

6. 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. 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 bin(30s)
```

Die Ergebnisse zeigen die Anzahl der Protokollereignisse in der Protokollgruppe, die von CloudWatch Logs für jeden Zeitraum von 30 Sekunden empfangen wurden.

5. 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).

Unterstützte Protokolle und erkannte Felder

CloudWatch Logs Insights unterstützt verschiedene Protokolltypen. Für jedes Protokoll, das an eine Protokollgruppe der Standardklasse 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.

> **Note**
>
> Die Felderkennung wird nur für Protokollgruppen der Standard-Protokollklasse unterstützt. Weitere Hinweise zu Protokollklassen finden Sie unterKlassen 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 Syntax der Logs Insights-Abfrage.

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.

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	Weitere Informationen finden Sie unter Felder in JSON-Protokollen.

Protokolltyp	Erkannte Protokollfelder
Protokolle im JSON-Format	
Andere Protokoll typen	`@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.

CloudWatch Syntax der Logs Insights-Abfrage

Mit CloudWatch Logs Insights verwenden Sie eine Abfragesprache, um Ihre Protokollgruppen abzufragen. Die Abfragesyntax unterstützt verschiedene Funktionen und Operationen, die unter anderem allgemeine Funktionen, Arithmetik- und Vergleichsoperationen sowie reguläre Ausdrücke beinhalten.

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

display	Zeigt ein bestimmtes Feld oder bestimmte Felder in den Abfrageergebnissen an.
fields	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.
filter	Filtert die Abfrage so, dass nur die Protokollereignisse zurückgegeben werden, die mindestens einer Bedingung entsprechen.
pattern	Gruppiert 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	Vergleicht 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	Extrahiert 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	Zeigt die zurückgegebenen Protokollereignisse in aufsteigender (`asc`) oder absteigender (`desc`) Reihenfolge an.
stats	Berechnet aggregierte Statistiken anhand der Werte in den Protokollfeldern.
limit	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.
dedup	Entfernt doppelte Ergebnisse auf der Grundlage bestimmter Werte in von Ihnen angegebenen Feldern.

`unmask`	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.
Weitere Vorgänge und Funktionen	CloudWatch Logs Insights unterstützt auch viele Vergleichs-, Arithmetik-, Datums- und Zahlenfunktionen, Zeichenketten, IP-Adressen und allgemeine Funktionen und Operationen.

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

Themen

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 von `display`, wenn Sie die verschiedenen von `fields` unterstützten Funktionen und Vorgänge zum Ändern von Feldwerten und zum Erstellen neuer Felder, die in Abfragen verwendet werden können, verwenden möchten.

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
```

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/
```

pattern

Verwenden Sie `pattern`, um Ihre Protokolldaten automatisch nach Mustern zu clustern.

Ein Muster ist eine gemeinsam genutzte Textstruktur, die sich in Ihren Protokollfeldern wiederholt. Sie können `pattern` sie 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.

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, wie z. B. eine Anforderungs-ID oder ein Zeitstempel,

werden durch <*> dargestellt. Zum Beispiel ist `[INFO] Request time: <*> 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

Verwenden Sie `parse`, um Daten aus einem Protokollfeld zu extrahieren und ein extrahiertes Feld zu erstellen, das Sie in Ihrer Abfrage verarbeiten können. **`parse`** unterstützt sowohl den globalen Modus mit Platzhaltern als auch reguläre Ausdrücke. Hinweise zur Syntax regulärer Ausdrücke finden Sie unterUnterstützte Syntax für reguläre Ausdrücke (Regex).

Sie können verschachtelte JSON-Felder mit einem regulären Ausdruck analysieren.

Beispiel: Analysieren eines verschachtelten JSON-Feld

Das Code-Snippet zeigt, wie ein JSON-Protokollereignis analysiert wird, das während der Aufnahme reduziert wurde.

```
{'fieldsA': 'logs', 'fieldsB': [{'fA': 'a1'}, {'fA': 'a2'}]}
```

Das Code-Snippet zeigt eine Abfrage mit einem regulären Ausdruck, der die Werte für `fieldsA` und `fieldsB` extrahiert, um die extrahierten Felder `fld` und `array` zu erstellen.

```
parse @message "'fieldsA': '*', 'fieldsB': ['*']" as fld, array
```

Benannte Erfassungsgruppen

Wenn Sie **`parse`** mit einem regulären Ausdruck verwenden, können Sie benannte Erfassungsgruppen verwenden, um ein Muster in einem Feld zu erfassen. Die Syntax lautet `parse @message (?<Name>pattern).`.

Im folgenden Beispiel wird eine Erfassungsgruppe für ein VPC-Flow-Protokoll verwendet, um die ENI in ein Feld namens `NetworkInterface` zu extrahieren.

```
parse @message /(?<NetworkInterface>eni-.*?) / display @timestamp, NetworkInterface
```

> **Note**
>
> JSON-Protokollereignisse werden während der Aufnahme reduziert. Derzeit wird das Parsen verschachtelter JSON-Felder mit einem Glob-Ausdruck nicht unterstützt. Sie können nur JSON-Protokollereignisse parsen, die nicht mehr als 200 Protokollereignisfelder enthalten. Wenn Sie verschachtelte JSON-Felder parsen, müssen Sie den regulären Ausdruck in Ihrer Abfrage so formatieren, dass er dem Format Ihres JSON-Protokollereignisses entspricht.

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
```

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.

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 und mit Zahlen beginnen, wird der Zahlenanteil zuerst verglichen. Werte, die mit derselben Anzahl von Ziffern beginnen, werden in aufsteigender Reihenfolge sortiert. Werte, die mit mehr Ziffern als andere Zeichenketten beginnen, werden nach den Werten sortiert, die mit weniger Ziffern beginnen. Wenn der Zahlenanteil zweier Werte identisch ist, werden sie auf der Grundlage der Unicode-Reihenfolge des restlichen Werts sortiert.
- Nichtnumerische Werte werden in Unicode-Reihenfolge sortiert, außer wenn die führenden Zeichen identisch sind und die nächsten Zeichen numerisch sind. In diesem Fall werden sie ähnlich wie in der vorherigen Logik sortiert: Werte mit derselben Anzahl von Ziffern nach den ersten führenden Zeichen werden in aufsteigender Reihenfolge sortiert. Dann werden Werte, die mehr Ziffern nach den führenden Zeichen haben, nach den Werten sortiert, die weniger Ziffern haben. Wenn der Zahlenanteil zweier Werte identisch ist, werden sie dann auf der Grundlage der Unicode-Reihenfolge der restlichen Werte sortiert.

Weitere Informationen zur Unicode-Reihenfolge finden Sie unter Liste der Unicode-Zeichen.

Das Folgende ist beispielsweise das Ergebnis einer Sortierung in aufsteigender Reihenfolge.

```
!: >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> sorted by unicode order
#
*%04
0# >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Alphanumeric starting with numbers
5A
111A   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  Starts with more digits than 5A, so it sorted
 to be later than 5A
2345_
```

```
@ >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 2345 is compared with @ in the unicode order,
@_
A >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Values starting with letters
A9876fghj
a12345hfh
0 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Number values
01
1
2
3
```

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.

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 bis zu zwei `stats`-Befehle in einer einzigen Abfrage verwenden. Auf diese Weise können Sie eine zusätzliche Aggregation für die Ausgabe der ersten Aggregation durchführen.

Beispiel: Abfrage mit zwei **`stats`**-Befehlen

Die folgende Abfrage ermittelt beispielsweise zuerst 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 zwei `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

Eine Abfrage kann maximal zwei `stats`-Befehle haben. Dieses Kontingent kann nicht geändert werden.

Wenn Sie einen `sort`- oder `limit`-Befehl verwenden, muss er nach dem zweiten `stats`-Befehl angezeigt werden. Wenn sie vor dem zweiten `stats`-Befehl steht, ist die Abfrage nicht gültig.

Wenn eine Abfrage zwei `stats`-Befehle enthält, werden die Teilergebnisse der Abfrage erst angezeigt, wenn die erste `stats`-Aggregation abgeschlossen ist.

Im zweiten `stats`-Befehl in einer einzelnen Abfrage können Sie nur auf Felder verweisen, die im ersten `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 dem ersten `stats`-Befehl verweisen, müssen in diesem ersten `stats`-Befehl definiert werden.

```
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 `bin` nicht im zweiten `stats`-Befehl verwenden können, ohne den ersten `stats`-Befehl zur Propagierung 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 ersten `stats`-Befehl, und dann können Sie es zusammen mit `dateceil` im zweiten `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	Beschreibung
`avg(fieldName: NumericLogField)`	Zahl	Der Mittelwert der Werte im angegebenen Feld.

`count()`
 	
`count(fieldName: LogField)`
	Zahl	Zählt die Protokollereignisse. `count()` (oder `count(*)`) zählt alle von der Abfrage zurückgegebenen Ereignisse, während `count(fieldName)` alle Datensätze, die den angegebenen Feldnamen enthalten, zählt.

`count_distinct(fieldName: LogField)`	Zahl	Liefert die Anzahl der eindeutigen Werte für das Feld. Wenn das Feld eine sehr hohe Kardinalität hat (zahlreiche eindeutige Werte enthält), ist der von `count_distinct` zurückgegebene Wert lediglich eine Annäherung.
`max(fieldName: LogField)`	LogFieldValue	Das Maximum der Werte für dieses Protokollfeld in den abgefragten Protokollen.
`min(fieldName: LogField)`	LogFieldValue	Das Minimum der Werte für dieses Protokollfeld in den abgefragten Protokollen.
`pct(fieldName: LogFieldValue, percent: number)`	LogFieldValue	Ein Perzentil gibt die relative Stelle eines Wertes in einer Datenmenge an. Zum Beispiel gibt `pct(@duration, 95)` den `@duration`-Wert zurück, bei dem 95 Prozent der Werte

Funktion	Ergebnistyp	Beschreibung
		von `@duration` niedriger als dieser Wert und 5 Prozent höher als dieser Wert sind.
`stddev(fieldName: NumericLogField)`	Zahl	Die Standardabweichung der Werte im angegebenen Feld.
`sum(fieldName: NumericLogField)`	Zahl	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	Beschreibung
`earliest(fieldName: LogField)`	LogField	Gibt den Wert von `fieldName` aus dem Protokollereignis mit dem frühesten Zeitstempel in den abgefragten Protokollen zurück.
`latest(fieldName: LogField)`	LogField	Gibt den Wert von `fieldName` aus dem Protokollereignis mit dem neuesten Zeitstempel in den abgefragten Protokollen zurück.
`sortsFirst(fieldName: LogField)`	LogField	Gibt den Wert von `fieldName` zurück, der in der Sortierung der abgefragten Protokolle an erster Stelle steht.
`sortsLast(fieldName: LogField)`	LogField	Gibt den Wert von `fieldName` 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.

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`.

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.

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

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	Beschreibung
`a + b`	Addition
`a - b`	Subtraktion

Operation	Beschreibung
`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 vonWAHRoderFALSCHaus.

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	Beschreibung
=	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	Beschreibung
`abs(a: number)`	Zahl	Absoluter Wert
`ceil(a: number)`	Zahl	Aufrunden (die kleinste ganze Zahl, die größer ist als der Wert von `a`).
`floor(a: number)`	Zahl	Abrunden (die größte ganze Zahl, die kleiner ist als der Wert von `a`).
`greatest(a: number, ...numbers: number[])`	Zahl	Liefert den größten Wert.
`least(a: number, ...numbers: number[])`	Zahl	Liefert den kleinsten Wert.
`log(a: number)`	Zahl	Natürlicher Logarithmus
`sqrt(a: number)`	Zahl	Quadratwurzel

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
- hstundenlang

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`s`. 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`s`. 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	Beschreibung
`bin(period: Period)`	Zeitstempel	Rundet den Wert von `@timestamp` auf den angegebenen Zeitraum und kürzt ihn dann.

Funktion	Ergebnistyp	Beschreibung

		`bin(5m)` rundet beispielsweise den Wert von `@timestamp` 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:
		
		`filter @message like /Exception/`
		`| stats count(*) as exceptionCount`
		`by bin(1h)`
		`| sort exceptionCount desc`
		
		Für die Funktion `bin` werden folgende Zeiteinheiten und Abkürzungen unterstützt. Alle Einheiten und Abkürzungen, die mehr als ein Zeichen enthalten, können durch Hinzufüge n 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`

Funktion	Ergebnistyp	Beschreibung
`datefloor(timestamp: Timestamp, period: Period)`	Zeitstempel	Kürzt den Zeitstempel auf den angegebenen Zeitraum. Zum Beispiel kürzt `datefloor(@timestamp, 1h)` alle Werte von `@timestamp` auf die letzte volle Stunde.
`dateceil(timestamp: Timestamp, period: Period)`	Zeitstempel	Rundet den Zeitstempel auf den angegebenen Zeitraum auf und kürzt ihn dann. Zum Beispiel kürzt `dateceil(@timestamp, 1h)` alle Werte von `@timestamp` auf die nächste volle Stunde.
`fromMillis(fieldName: number)`	Zeitstempel	Interpretiert das Eingabefeld als die Anzahl der Millisekunden seit der Unix-Epoche und konvertiert es in einen Zeitstempel.
`toMillis(fieldName: Timestamp)`	Zahl	Konvertiert den im benannten Feld gefundenen Zeitstempel in eine Zahl, die die Millisekunden seit der Unix-Epoche darstellt. Beispiel: `toMillis(@timestamp)` konvertiert den Zeitstempel `2022-01-14T13:18:031.000-08:00` zu `1642195111000`.

> **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	Beschreibung
`ispresent(fieldName: LogField)`	Boolesch	Gibt `true` zurück, wenn das Feld existiert.
`coalesce(fieldName: LogField, ...fieldNames: LogField[])`	LogField	Liefert den ersten Nicht-Null-Wert aus der Liste.

IP-Adressenzeichenfolgen

IP-Adressenzeichenfolgen

Sie können IP-Adressen-Zeichenfolgenfunktionen in den Befehlen `filter` und `fields` sowie als Argumente für andere Funktionen verwenden.

Funktion	Ergebnistyp	Beschreibung
`isValidIp(fieldName: string)`	boolesch	Gibt `true` zurück, wenn das Feld eine gültige IPv4- oder IPv6-Adresse ist.
`isValidIpV4(fieldName: string)`	boolesch	Gibt `true` zurück, wenn das Feld eine gültige IPv4-Adresse ist.
`isValidIpV6(fieldName: string)`	boolesch	Gibt `true` zurück, wenn das Feld eine gültige IPv6-Adresse ist.
`isIpInSubnet(fieldName: string, subnet: string)`	boolesch	Gibt `true` 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 `192.0.2.0/24` oder `2001:db8::/32` , wobei `192.0.2.0` oder `2001:db8::` der Anfang des CIDR-Blocks ist.
`isIpv4InSubnet(fieldName: string, subnet: string)`	boolesch	Gibt `true` zurück, wenn das Feld eine gültige IPv4-Adresse innerhalb des angegebenen v4-Subnetzes ist. Verwenden Sie bei der

Funktion	Ergebnistyp	Beschreibung
		Angabe des Subnetzes die CIDR-Notation wie `192.0.2.0/24` , wobei `192.0.2.0` der Anfang des CIDR-Blocks ist.
`isIpv6InSubnet(fieldName: string, subnet: string)`	boolesch	Gibt `true` 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 `2001:db8::/32` , wobei `2001:db8::` 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	Beschreibung
`isempty(fieldName: string)`	Zahl	Gibt 1 zurück, wenn das Feld fehlt oder eine leere Zeichenkette ist.
`isblank(fieldName: string)`	Zahl	Gibt 1 zurück, wenn das Feld fehlt, eine leere Zeichenkette ist oder nur Leerzeichen enthält.
`concat(str: string, ...strings: string[])`	Zeichenfolge	Verkettet die Zeichenketten.

`ltrim(str: string)`
 	
`ltrim(str: string, trimChars: string)`
	Zeichenfolge	Wenn die Funktion kein zweites Argument hat, entfernt sie die Whitespaces von der linken Seite der Zeichenfolge.

Funktion	Ergebnistyp	Beschreibung
		Wenn die Funktion ein zweites Zeichenfolgen-Argument hat, entfernt sie keine Whitespac es. Stattdessen entfernt sie die Zeichen in `trimChars` links von `str`. Beispielsweise gibt `ltrim("xyZxyfooxyZ","xyZ")` `"fooxyZ"` zurück.

`rtrim(str: string)`
 	
`rtrim(str: string, trimChars: string)`
	Zeichenfolge	Wenn die Funktion kein zweites Argument hat, entfernt sie die Whitespaces von der rechten Seite der Zeichenfo lge. Wenn die Funktion ein zweites Zeichenfolgen-Argu ment hat, entfernt sie keine Whitespaces. Stattdessen entfernt sie die Zeichen von `trimChars` rechts von `str`. Beispielsweise gibt `rtrim("xyZfooxyxyZ","xyZ")` `"xyZfoo"` zurück.

Funktion	Ergebnistyp	Beschreibung

`trim(str: string)`
 	
`trim(str: string, trimChars: string)`
	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 `trimChars` von beiden Seiten von `str`. Beispielsweise gibt `trim("xyZ xyfooxyxyZ","xyZ")` `"foo"` zurück.

`strlen(str: string)`	Zahl	Liefert die Länge der Zeichenkette in Unicode-Codepunkten.
`toupper(str: string)`	Zeichenfolge	Konvertiert die Zeichenkette in Großbuchstaben.
`tolower(str: string)`	Zeichenfolge	Konvertiert die Zeichenkette in Kleinbuchstaben.

Funktion	Ergebnistyp	Beschreibung

`substr(str: string, startIndex: number)`
 	
`substr(str: string, startIndex: number, length: number)`
	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 `substr("xyZfooxyZ",3, 3)` `"foo"` zurück.

`replace(fieldName: string, searchValue: string, replaceValue: string)`	Zeichenfolge	Ersetzt alle Instances von `searchValue` in `fieldName: string` mit `replaceValue` . Beispiel: Die Funktion `replace(logGroup,"smoke_test","Smoke")` sucht nach Protokollereignissen, bei denen das Feld `logGroup` den Zeichenfolgenwert `smoke_test` enthält und den Wert durch die Zeichenfolge `Smoke` ersetzt.
`strcontains(str: string, searchValue: string)`	Zahl	Gibt 1 zurück, wenn `str` `searchValue` enthält; ansonsten 0.

Felder, die Sonderzeichen enthalten

Sie müssen Protokollfelder, die in Abfragen benannt werden, die andere Zeichen als das @-Symbol, den Punkt (`.`) und nicht-alphanumerische Zeichen enthalten, in Backtick-Schlüsseln (`` ` ``) einschließen.

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`um 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
```

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:

```
<*> <*> [INFO] Calling DynamoDB to store for resource id <*>
```

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 dadurch dargestellt, dass es <*> in CloudWatch Logs angezeigt wird.

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 finden 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 Logs, 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 mit der Abfrage übereinstimmten, ähnlich der Meldung 1000 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 1000 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<*>.
   - 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 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 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.

    Der Bereich Pattern Inspect 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 habenpattern, weil eine Abfrage nicht gültig ist, wenn sie nach einem `pattern` Befehl einen `sort` Befehl enthält. Es ist gültig, ein `pattern` vor einem zu habensort.

  Weitere Informationen zur `pattern` Syntax finden Sie unterpattern.
- 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 einfügen, 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 Siepattern `@fieldname`.
- Abfragen, bei denen es sich nicht um eine Protokollausgabe handelt, wie z. B. Abfragen mit dem `stats` Befehl, geben keine Musterergebnisse zurück.

Vergleiche (Diff) mit früheren Zeitbereichen

Sie können CloudWatch Logs Insights verwenden, um Änderungen an Ihren Protokollereignissen 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 unterMusteranalyse.

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.
- Vorwoche — 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 CloudWatch Amazon-Preise.

Um eine Vergleichsabfrage auszuführen

1. Öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im Navigationsbereich Logs, Logs Insights aus.

   Im Abfragefeld wird eine Standardabfrage angezeigt.
3. Behalten Sie die Standardabfrage bei oder geben Sie eine andere Abfrage ein.
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. Die Standardabfrage bezieht sich auf die Protokolldaten der letzten Stunde.
6. 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.
7. Wählen Sie Abfrage ausführen.

   Damit die Abfrage die Daten aus dem Vergleichszeitraum abruft, wird der `diff` Befehl an Ihre Abfrage angehängt.
8. 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<*>. 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`

9. 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.

Beispielabfragen

Dieser Abschnitt enthält eine Liste allgemeiner und nützlicher Abfragebefehle, die Sie in der CloudWatch Konsole ausführen können. Informationen zur Ausführung eines Abfragebefehls finden Sie unter Tutorial: Eine Beispielabfrage ausführen und ändern im Amazon CloudWatch Logs-Benutzerhandbuch.

Weitere Informationen zur Abfragesyntax finden Sie unterCloudWatch Syntax der Logs Insights-Abfrage.

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 provisonedMemoryMB,
        min(@maxMemoryUsed / 1000 / 1000) as smallestMemoryRequestMB,
        avg(@maxMemoryUsed / 1000 / 1000) as avgMemoryUsedMB,
        max(@maxMemoryUsed / 1000 / 1000) as maxMemoryUsedMB,
        provisonedMemoryMB - 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

Ermittelt die Anzahl der Protokolleinträge pro Service, 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
```

Erstellen Sie 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 Beitragszahler 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
```

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.

Speichern Sie Logs Insights-Abfragen und führen Sie sie erneut CloudWatch 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.

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.

Eine Abfrage 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.

   Wenn Sie die Schaltfläche Speichern nicht sehen, müssen Sie zum neuen Design der CloudWatch Logs-Konsole wechseln. Gehen Sie hierzu wie folgt vor:

   a. Wählen Sie im Navigationsbereich Protokollgruppen aus.
   b. Wählen Sie Try the new design (Neues Design testen) aus.
   c. Wählen Sie im Navigationsbereich Insights (Einsichten) aus und kehren Sie zu Schritt 3 dieses Verfahrens zurück.

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. Wählen Sie Speichern.

> **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.

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 Saved queries (Gespeicherte Abfragen) aus. Anschließend wird sie im Abfrage-Editor angezeigt.
5. Wählen Sie Ausführen aus.

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 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.
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.

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 30 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 30 gleichzeitige CloudWatch Logs Insights-Abfragen pro Konto beschränkt, einschließlich Abfragen, die zu Dashboards hinzugefügt wurden.

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 die 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 wählen, ob Sie stattdessen einen AWS KMS Schlüssel verwenden möchten, um diese Ergebnisse 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 Ausgabebeispiel:

```
{
    "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 `policy.json` mit dem folgenden get-key-policyBefehl:

```
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 die CloudWatch Logs Insights-Abfrageergebnisse im angegebenen Konto.

```
{
 "Version": "2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::account_ID: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:region:account_ID:query-result:*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "Your_account_ID"
                }
            }
        }
    ]
}
```

Fügen Sie abschließend die aktualisierte Richtlinie mit dem folgenden put-key-policyBefehl hinzu:

```
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 disassociate-kms-key-Befehl wie folgt:

```
aws logs associate-kms-key --resource-identifier "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

Verwenden Sie den folgenden disassociate-kms-keyBefehl, um die Zuordnung des KMS-Schlüssels zu trennen, der den Abfrageergebnissen zugeordnet ist:

```
aws logs disassociate-kms-key --resource-identifier "arn:aws:logs:region:account-id:query-result:*"
```

Verwenden Sie natürliche Sprache, um CloudWatch Logs Insights-Abfragen zu generieren und zu aktualisieren

Diese Funktion befindet sich in den Regionen USA Ost (Nord-Virginia), USA West (Oregon) und Asien-Pazifik (Tokio) als Vorschauversion für CloudWatch Logs und kann sich ändern.

CloudWatch Logs unterstützt eine Abfragefunktion in natürlicher Sprache, mit der Sie Abfragen für CloudWatch Logs Insights und CloudWatch Metrics Insights generieren und 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 in natürlicher Sprache generiert eine Abfrage auf der Grundlage einer Eingabeaufforderung, die Sie eingeben, und bietet eine line-by-line Erläuterung der Funktionsweise der Abfrage. 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.“

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 aus.

Important

Um die Abfragefunktion in natürlicher Sprache verwenden zu können, müssen Sie die ReadOnlyAccessRichtlinie CloudWatchLogsFullAccessCloudWatchLogsReadOnlyAccess, AdministratorAccess, oder verwenden.

Sie können die `cloudwatch:GenerateQuery`-Aktion auch in eine neue oder bestehende, vom Kunden verwaltete oder integrierte Richtlinie aufnehmen.

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-Abfrageeditor und zur Syntax finden Sie unter CloudWatch Logs Insights-Abfragesyntax.

Beispiel: Eine Abfrage in natürlicher Sprache generieren

Um eine Abfrage in natürlicher Sprache zu generieren, geben Sie eine Aufforderung ein und wählen Sie Neue Abfrage generieren. Dieses Beispiel zeigt eine Abfrage, die eine einfache Suche durchführt.

Telefonansage

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
```

Abfrage

Im Folgenden finden Sie ein Beispiel für eine Abfrage, die die Funktion natürlicher Sprache anhand der Eingabeaufforderung generiert. 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.

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. 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 Logs Insights zu verbessern. CloudWatch 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 .

Erkennung von Protokollanomalien

Sie können für jede Protokollgruppe einen Detektor für Protokollanomalien erstellen. Der Anomaliedetektor scannt die in die Protokollgruppe aufgenommenen Protokollereignisse und findet Anomalien in den Protokolldaten. Die Anomalieerkennung nutzt maschinelles Lernen und Mustererkennung, um Basiswerte für typische Protokollinhalte zu erstellen.

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 beispielsweise das folgende Beispiel mit drei Protokollereignissen an.

```
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:

```
<*> <*> [INFO] Calling DynamoDB to store for resource id <*>
```

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. Dynamische Token werden dargestellt<*>, wenn CloudWatch Logs das Muster anzeigt. Jeder unterschiedliche Wert, der für ein dynamisches Token gefunden wird, wird als Tokenwert bezeichnet.

Zu den häufigsten Beispielen für dynamische Token gehören Fehlercodes, Zeitstempel und Anforderungs-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.

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 ListAnomaliesAPI-Vorgang zurückgegeben wird. Wenn diese Zeitspanne für eine Anomalie verstrichen ist und sie weiterhin auftritt, wird sie automatisch als normales Verhalten akzeptiert, und das Anomaliedetektormodell markiert 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 unterMusteranalyse.

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.

Aktivieren Sie die Erkennung von Anomalien 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

So erstellen Sie einen Detektor für Protokollanomalien

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, kann es angemessen sein, 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 im Feld Maximaler Zeitraum für die Sichtbarkeit von Anomalien (Tage) einen neuen Wert 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 von all Ihren Protokollanomaliedetektoren gefundenen Anomalien einzusehen

1. Öffnen Sie die CloudWatch Konsole unter 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. 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 unterErkennung von Protokollanomalien.
   - Beim Trend im Anomalieprotokoll wird ein Histogramm angezeigt, 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 unterErkennung von Protokollanomalien.
   - Beim Trend im Anomalieprotokoll wird ein Histogramm angezeigt, 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 aus.

   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 aus.

   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 immer es ***`your-metric-name`***ist. , 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 Anomaliedetektor 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 `ALARM``OK`, 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 mithilfe eines AWS KMS Schlüssels. 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 Ausgabebeispiel:

```
{
    "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-policyBefehl:

```
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.

```
{
  "Version": "2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::Your_account_ID:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.REGION.amazonaws.com"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "ArnLike": {
          "kms:EncryptionContext:aws:logs:arn":
 "arn:aws:logs:REGION:Your_account_ID:anomaly-detector:*"
        }
      }
    },
    {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.REGION.amazonaws.com"
        },
```

```
        "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
        ],
        "Resource": "*",
        "Condition": {
            "ArnLike": {
                "kms:EncryptionContext:aws-crypto-ec:aws:logs:arn":
 "arn:aws:logs:REGION:Your_account_ID:anomaly-detector:*"
            }
        }
    }
  ]
}
```

Fügen Sie abschließend die aktualisierte Richtlinie mit dem folgenden put-key-policyBefehl 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 der AWS CLI APIs oder erstellen.

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.

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.

Verwenden Sie die Verfahren in diesem Abschnitt für die Arbeit mit Log-Gruppen und Log-Streams.

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 Protokollgruppen 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 mit einer der folgenden Methoden auch andere Protokollereignisse 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-eventsLädt Stapel von Protokollereignissen in Logs hoch. CloudWatch
- Programmgesteuert — Mit der PutLogEventsAPI 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 groups (Protokollgruppen) 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.

Verwenden von Live Tail zur Anzeige von Protokollen in nahezu Echtzeit

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.

> **Note**
>
> Live Tail wird nur für Protokollgruppen der Standard-Protokollklasse unterstützt. Weitere Informationen zu Protokollklassen finden Sie unterKlassen protokollieren.

In den folgenden Abschnitten wird erklärt, wie Sie Live Tail in der Konsole verwenden. 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.

Starten einer Live-Tail-Sitzung

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.

> **Note**
>
> 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.

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.
   - Ereignisse/Sek. zeigt an, wie viele erfasste Protokollereignisse pro Sekunde mit den von Ihnen festgelegten Filtern übereinstimmen.
   - Um zu verhindern, dass die Sitzung zu schnell scrollt, weil viele Ereignisse den Filtern entsprechen, werden in den CloudWatch Protokollen möglicherweise nur einige übereinstimmende Ereignisse angezeigt. Wenn dies der Fall ist, wird der Prozentsatz der übereinstimmenden Ereignisse, die auf dem Bildschirm angezeigt werden, in % angezeigt.

8. Um den Fluss der Ereignisse anzuhalten und zu untersuchen, was gerade angezeigt wird, klicken Sie auf 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.

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 gefunden 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 CloudWatch kontenübergreifende Beobachtbarkeit.

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 groups (Protokollgruppen) 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 groups (Protokollgruppen) 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 mit dem AWS CLI

Sie können mit dem nach Protokolleinträgen suchen, die bestimmte Kriterien erfüllen AWS CLI.

Um Protokolleinträge mit dem zu suchen AWS CLI

Führen Sie in einer Befehlszeile den folgenden filter-log-eventsBefehl 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 den AWS CLI

Führen Sie in einer Befehlszeile den folgenden filter-log-eventsBefehl 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 Protokollgruppen aus.
3. Gehen Sie zur Protokollgruppe, die aktualisiert werden soll.
4. Wählen Sie in der Spalte Expire Events After für diese Protokollgruppe die aktuelle Einstellung für die Aufbewahrung, wie z. B. Never Expire.
5. Wählen Sie im Dialogfeld Edit Retention (Aufbewahrung bearbeiten) in Retention (Aufbewahrung) einen Wert für die Protokollaufbewahrung aus. Wählen Sie anschließend OK aus.

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)

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 für die Protokolldaten im Ruhezustand. Als Alternative können Sie AWS Key Management Service für diese Verschlüsselung verwenden. Wenn Sie dies tun, erfolgt die Verschlüsselung mithilfe eines AWS KMS Schlüssels. 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, der `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.

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 ihres gesamten Aufbewahrungszeitraums 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 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 Ausgabebeispiel:

```
{
    "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 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 Protokollgruppen zu beschränken.

Speichern Sie zunächst die Standardrichtlinie für Ihren KMS-Schlüssel `policy.json` mit dem folgenden get-key-policyBefehl:

```
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 schränkt den Schlüssel auf einen einzelnen Protokollgruppen-ARN ein.

```
{
 "Version": "2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::Your_account_ID: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": {
                    "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:region:account-
id: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.

```
{
    "Version": "2012-10-17",
    "Id": "key-default-1",
    "Statement": [
```

```
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::Your_account_ID: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": {
                "ArnLike": {
                    "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:region:account-
id:*"
                }
            }
        }
    ]
}
```

Fügen Sie abschließend die aktualisierte Richtlinie mit dem folgenden put-key-policyBefehl hinzu:

```
aws kms put-key-policy --key-id key-id --policy-name default --policy file://
policy.json
```

Schritt 3: Verknüpfen eines KMS-Schlüssels mit einer Protokollgruppe

Sie können einen KMS-Schlüssel während oder nach der Erstellung einer Protokollgruppe mit dieser verknüpfen.

Verwenden Sie den folgenden describe-log-groupsBefehl, 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 4: Aufheben der Verknüpfung eines Schlüssels mit einer Protokollgruppe

Verwenden Sie den folgenden disassociate-kms-keyBefehl, 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, verwendet CloudWatch Logs jetzt Protokollgruppen-ARNs als Teil des Verschlüsselungskontextes, 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, IAM-Richtlinienbedingungen zu verwenden, um 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:

   ```
   {
       "Version": "2012-10-17",
       "Id": "key-default-1",
       "Statement": [
           {
               "Sid": "Enable IAM User Permissions",
               "Effect": "Allow",
   ```

```
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT-ID: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": {
                "ArnLike": {
                    "kms:EncryptionContext:aws:logs:arn":
 "arn:aws:logs:REGION:ACCOUNT-ID: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.

```
{
    "Version": "2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::Your_account_ID: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 nur für Protokollgruppen der Standard-Protokollklasse unterstützt. Wenn Sie eine Datenschutzrichtlinie für alle Protokollgruppen in Ihrem Konto erstellen, gilt diese nur für Protokollgruppen der Standard-Protokollklasse. Weitere Informationen zu Protokollklassen finden Sie unterKlassen 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 Alarme 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 unterÜberwachung 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 analyisieren 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

Themen

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 Datenschutzrichtlinie ohne Audit-Ziele	`logs:PutAccountPolicy`	*
	`logs:PutDataProtectionPolicy`	*
Erstellen Sie eine Datenschutzrichtlinie mit CloudWatch Logs als Auditziel	`logs:PutAccountPolicy`	*
	`logs:PutDataProtectionPolicy`	*
	`logs:CreateLogDelivery`	*
	`logs:PutResourcePolicy`	*

Operation	IAM-Berechtigung erforderlich	Ressource
	`logs:DescribeResourcePolicies`	*
	`logs:DescribeLogGroups`	*
Erstellen Sie eine Datenschutzrichtlinie mit Firehose als Auditziel	`logs:PutAccountPolicy`	*
	`logs:PutDataProtectionPolicy`	*
	`logs:CreateLogDelivery`	*
	`firehose:TagDeliveryStream`	`arn:aws:logs:::deliverystream/` *YOUR_DELIVERY_STREAM*
Erstellen einer Datenschutzrichtlinie mit Amazon S3 als Audit-Ziel	`logs:PutAccountPolicy`	*
	`logs:PutDataProtectionPolicy`	*
	`logs:CreateLogDelivery`	*
	`s3:GetBucketPolicy`	`arn:aws:s3:::` *YOUR_BUCKET*
	`s3:PutBucketPolicy`	`arn:aws:s3:::` *YOUR_BUCKET*

Operation	IAM-Berechtigung erforderlich	Ressource
Aufheben der Maskierung für Protokollereignisse in einer angegebenen Protokollgruppe	`logs:Unmask`	`arn:aws:logs:::log-group:*`
Anzeigen einer bestehenden Datenschutzrichtlinie	`logs:GetDataProtectionPolicy`	*
Löschen einer Datenschutzrichtlinie	`logs:DeleteAccountPolicy`	*
	`logs:DeleteDataProtectionPolicy`	*

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 Datenschutzrichtlinie ohne Audit-Ziele	`logs:PutDataProtectionPolicy`	`arn:aws:logs:::log-group:` *YOUR_LOG_GROUP* `:*`

Operation	IAM-Berechtigung erforderlich	Ressource

Erstellen Sie eine Datenschutzrichtlinie mit CloudWatch Logs als Auditziel	`logs:PutDataProtectionPolicy`
	`logs:CreateLogDelivery`
	`logs:PutResourcePolicy`
	`logs:DescribeResourcePolicies`
	`logs:DescribeLogGroups`
		`arn:aws:logs:::log-group:` *YOUR_LOG_GROUP* `:*`
		`*`
		`*`
		`*`
		`*`

Erstellen Sie eine Datenschutzrichtlinie mit Firehose als Auditziel	`logs:PutDataProtectionPolicy`
	`logs:CreateLogDelivery`
	`firehose:TagDeliveryStream`
		`arn:aws:logs:::log-group:` *YOUR_LOG_GROUP* `:*`
		`*`
		`arn:aws:logs:::deliverystream/` *YOUR_DELIVERY_STREAM*

Erstellen einer Datenschutzrichtlinie mit Amazon S3 als ein Audit-Ziel	`logs:PutDataProtectionPolicy`
	`logs:CreateLogDelivery`
	`s3:GetBucketPolicy`
	`s3:PutBucketPolicy`
		`arn:aws:logs:::log-group:` *YOUR_LOG_GROUP* `:*`
		`*`
		`arn:aws:s3:::` *YOUR_BUCKET*
		`arn:aws:s3:::` *YOUR_BUCKET*

Operation	IAM-Berechtigung erforderlich	Ressource
Entfernen von Maskierungen bei Protokollereignissen	`logs:Unmask`	`arn:aws:logs:::log-group:` *YOUR_LOG_GROUP* `:*`
Eine bestehende Datenschutzrichtlinie ansehen	`logs:GetDataProtectionPolicy`	`arn:aws:logs:::log-group:` *YOUR_LOG_GROUP* `:*`
Löschen einer Datenschutzrichtlinie	`logs:DeleteDataProtectionPolicy`	`arn:aws:logs:::log-group:` *YOUR_LOG_GROUP* `:*`

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.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "YOUR_SID_1",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:PutResourcePolicy",
                "logs:DescribeLogGroups",
                "logs:DescribeResourcePolicies"
            ],
            "Resource": "*"
        },
        {
            "Sid": "YOUR_SID_2",
```

```
            "Effect": "Allow",
            "Action": [
                "logs:GetDataProtectionPolicy",
                "logs:DeleteDataProtectionPolicy",
                "logs:PutDataProtectionPolicy",
                "s3:PutBucketPolicy",
                "firehose:TagDeliveryStream",
                "s3:GetBucketPolicy"
            ],
            "Resource": [
                "arn:aws:firehose:::deliverystream/YOUR_DELIVERY_STREAM",
                "arn:aws:s3:::YOUR_BUCKET",
                "arn:aws:logs:::log-group:YOUR_LOG_GROUP:*"
            ]
        }
    ]
}
```

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 Datenbezeichner 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 `DataIdentifer`-Array als auch eine `Operation`-Eigenschaft mit einer`Audit`-Aktion enthalten. Das `DataIdentifer`-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 `DataIdentifer`-Array als auch eine `Operation`-Eigenschaft mit einer `Deidentify`-Aktion enthalten. Das `DataIdentifer`-Array muss genau mit dem `DataIdentifer`-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 den Vereinigten Staaten.

Informationen zu Richtlinien, die benutzerdefinierte Datenkennungen angeben, finden Sie unterVerwenden 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

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 `DataIdentifer`-Array als auch eine `Operation`-Eigenschaft mit einer`Audit`-Aktion enthalten. Das `DataIdentifer`-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 `DataIdentifer`-Array als auch eine `Operation`-Eigenschaft mit einer `Deidentify`-Aktion enthalten. Das `DataIdentifer`-Array muss genau mit dem `DataIdentifer`-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 unterCloudWatch Syntax der Logs Insights-Abfrage.
- Verwenden Sie eine GetLogEvents FilterLogEventsOder-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

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.

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
`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. z. B. `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
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
HealthInsuranceClaimNumber	Gesundheit	US
HealthInsuranceNumber	Gesundheit	FR
HealthcareProcedureCode	Gesundheit	US
IndividualTaxIdentificationNumber	Persönlich	US
InseeCode	Persönlich	FR

Kennung	Kategorie	Länder und Sprachen
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
Ssn	Persönlich	ES, US
TaxId	Persönlich	DE, ES, FR, GB
ZipCode	Persönlich	US

Anmeldeinformationen

CloudWatch In Logs Data Protection können die folgenden Arten von Anmeldeinformationen gefunden werden.

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen
AWS geheimer Zugriffss chlüssel	`AwsSecretKey`	`aws_secret_access_key` , `credentials` , `secret access key`, `secret key`, `set-awscredential`	Alle
Privater OpenSSH-Schlüssel	`OpenSSHPrivateKey`	None	Alle
Privater PGP-Schlüssel	`PgpPrivateKey`	None	Alle
Privater Pkcs-Schlüssel	`PkcsPrivateKey`	None	Alle
Privater PuTTY-Schlüssel	`PuttyPrivateKey`	None	Alle

Datenkennungs-ARNs für Anmeldeinformations-Datentypen

Im Folgenden werden die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgelistet, die Sie Ihren Datenschutzrichtlinien hinzufügen können.

ARNs der Datenkennung für Anmeldeinformationen
`arn:aws:dataprotection::aws:data-identifier/AwsSecretKey`
`arn:aws:dataprotection::aws:data-identifier/OpenSshPrivateKey`
`arn:aws:dataprotection::aws:data-identifier/PgpPrivateKey`

ARNs der Datenkennung für Anmeldeinformationen
`arn:aws:dataprotection::aws:data-identifier/PkcsPrivateKey`
`arn:aws:dataprotection::aws:data-identifier/PuttyPrivateKey`

Gerätekennungen

CloudWatch Der Datenschutz protokolliert die folgenden Typen von Gerätekennungen.

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen
IP-Adresse	`IpAddress`	None	Alle

Datenkennungs-ARNs für Gerätedatentypen

Im Folgenden werden die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgelistet, die Sie Ihren Datenschutzrichtlinien hinzufügen können.

Gerätedatenkennungs-ARN
`arn:aws:dataprotection::aws:data-identifier/IpAddress`

Finanzinformationen

CloudWatch In 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 Regione	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.	Frankreich, Deutschland, Italien, Spanien, Großbritannien	Dazu gehören Internationale Bankkontonummern (IBANs), die aus bis zu 34 alphanumerischen Zeichen bestehen, einschließlich Elementen wie dem Ländercode.
Ablaufdatum der Kreditkarte	`CreditCardExpiration`	`exp d`, `exp m`, `exp y`, `expiration` , `expiry`	Alle	
Kreditkartennummer	`CreditCardNumber`	`account number`, `american express`, `amex`, `bank card`, `card`, `card number`, `card num`, `cc #`, `ccn`, `check card`, `credit`, `credit card#`, `dankort`, `debit`,	Alle	Für die Erkennung müssen die Daten eine 13—

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regione	Hinweise
		`debit card`, `diners club`, `discover`, `electron`, `japanese card bureau`, `jcb`, `mastercard` , `mc`, `pan`, `payment account number`, `payment card number`, `pcn`, `union pay`, `visa`		19-stellige Sequenz sein, die der Luhn-Scheckformel entspricht und ein Standard-Kartennummernpräfix für jede der folgenden Arten von Kreditkarten verwendet: American Express, Dankort, Diner's Club, Discover, Electron, Japanese Card

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regione	Hinweise
				Bureau (JCB) UnionPay, Mastercard und Visa.
Bestätigungscode für die Kreditkarte	`CreditCardSecurityCode`	`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, um Internationale Bankkontonummern (IBANs) zu erkennen, die aus bis zu 34 alphanumerischen Zeichen bestehen, einschließlich Elementen wie dem Ländercode.

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`

Land	Schlüsselwörter
Deutschland	account code, account number, accountno# , accountnumber# , bankleitzahl , bban, customer account id, customer account number, customer bank account id, geheimzahl , iban, kartennummer , 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.

```
122000000000003, 2222405343248877, 2222990905257051, 2223007648726984,
 2223577120017656,
30569309025904, 34343434343434, 3528000700000000, 3530111333300000, 3566002020360505,
 36148900647913,
36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237,
 4012888888881881,
4111111111111111, 4222222222222, 4444333322221111, 4462030000000000, 4484070000000000,
 4911830000000,
4917300800000000, 4917610000000000, 4917610000000000003, 5019717010103742,
 5105105105105100,
```

```
5111010030175156, 5185540810000019, 5200828282828210, 5204230080000017,
 5204740009900014, 5420923878724339,
5454545454545454, 5455330760000018, 5506900490000436, 5506900490000444,
 5506900510000234, 5506920809243667,
5506922400634930, 5506927427317625, 5553042241984105, 5555553753048194,
 5555555555554444, 5610591081018250,
6011000990139424, 6011000400000000, 6011111111111117, 630490017740292441,
 630495060000000000,
6331101999990016, 6759649826438453, 6799990100000000019, and 76009244561.
```

Datenkennungs-ARNs für Finanzdatentypen

Im Folgenden werden die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgelistet, die Sie Ihren Datenschutzrichtlinien hinzufügen können.

ARNs der Finanzdatenkennungen
`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 Logs Der Datenschutz kann die folgenden Arten von geschützten Gesundheitsinformationen (PHI) finden.

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`, `numero conto medico`, `numéro d’assurance maladie` , `numéro de carte d’assuran`

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen
		ce , numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin , sairausvakuutuskortti , sairausvakuutusnumero , sjukförsäkring nummer, sjukförsäkringskort , suomi ehic-numero , tarjeta de salud, terveyskortti , tessera sanitaria assicurazione numero , versicherungsnummer	
Health Insurance Claim Number (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#, hicno#	Vereinigte Staaten
Krankenversicherungs- oder medizinische Identifizierungsnummer	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	Frankreich
Standardisierte Codes für medizinische Leistungen (HCPCS)	HealthcareProcedureCode	current procedural terminology , hcpcs, healthcare common procedure coding system	Vereinigte Staaten

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regionen
Medicare-Versicherungsnummer (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

Datenkennungs-ARNs für geschützte Gesundheitsdatentypen (PHI)

Im Folgenden werden die Amazon-Ressourcennamen (ARNs) für die Datenkennungen aufgelistet, die Sie Ihren Datenschutzrichtlinien zu geschützten Gesundheitsdaten hinzufügen können.

ARNS für PHI-Datenkennungen
`arn:aws:dataprotection::aws:data-identifier/DrugEnforcementAgencyNumber-US`
`arn:aws:dataprotection::aws:data-identifier/HealthcareProcedureCode-US`

ARNS für PHI-Datenkennungen
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 Logs Data Protection kann die folgenden Arten von persönlich identifizierbaren Informationen (PII) finden.

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 Regione	Hinweise
Geburtsdatum	`DateOfBirth`	`dob`, `date of birth`, `birthdate` , `birth date`, `birthday`, `b-day`, `bday`	Any	Der Support umfasst die meisten Datumsformate, z. B. nur Ziffern und Kombinationen aus Ziffern und Monatsnamen. Datumskomponenten können durch Leerzeichen, Schrägstriche (/) oder Bindestriche (-) getrennt werden.

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regione	Hinweise
Código de Endereçamento Postal (CEP)	`CepCode`	`cep`, `código de endereçamento postal`, `codigo de endereçamento postal`	Brasilien	
Cadastro Nacional da Pessoa Jurídica (CNPJ)	`Cnpj`	`cadastro nacional da pessoa jurídica`, `cadastro nacional da pessoa juridica`, `cnpj`	Brasilien	
Cadastro de Pessoas Físicas (CPF)	`CpfCode`	`Cadastro de pessoas fisicas`, `cadastro de pessoas físicas`, `cadastro de pessoa física`, `cadastro de pessoa fisica`, `cpf`	Brasilien	

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regione	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	`ElectoralRollNumber`	`electoral #`, `electoral number`, `electoral roll #`, `electoral roll no.`, `electoral roll number`, `electoral rollno`	Großbritannien und Nordirland	

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regione	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 Regione	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#` , `nationalinsurancenumber` , `nin`, `nino`	Großbritannien und Nordirland	–

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regione	Hinweise
Número de identidad de extranjero (NIE)	`NieNumber`	Ja. Für verschiedene Länder gelten verschied ene Schlüsselwörter. Einzelheiten finden Sie in der Tabelle Individua l taxpayer identification numbers (Individuelle Steuerzahler-Identifikation snummern) weiter unten in diesem Abschnitt.	Spanien	
Número de Identificación Fiscal (NIF)	`NifNumber`	Ja. Für verschiedene Länder gelten verschied ene Schlüsselwörter. Einzelheiten finden Sie in der Tabelle Individua l taxpayer identification numbers (Individuelle Steuerzahler-Identifikation snummern) weiter unten in diesem Abschnitt.	Spanien	
Passnummer	`PassportN umber`	Ja. Für verschiedene Länder gelten verschied ene Schlüsselwörter. Einzelheiten finden Sie in der Tabelle Keywords for passport numbers (Schlüsselwörter für Reisepassnummern) weiter unten in diesem Abschnitt.	Kanada, Frankrei h, Deutschl nd, Italien, Spanien, Großbrit nnien, USA	

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regione	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 Regione	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 Landesvorwahl enthalten. Wenn sich ein Schlüsselwort

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regione	Hinweise
				nicht in der Nähe der Daten befindet, muss die Nummer eine Landesvorwahl enthalten.
Postal Code (Postleitzahl)	`PostalCode`	None	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`, `socialsecurityno#` , `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 Regione	Hinweise
Zip code (Postleitzahl)	`ZipCode`	`zip code`, `zip+4`	Vereinigte Staaten	Postleitzahl der USA.

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regione	Hinweise
Postanschrift	`Address`	None	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`	None	Any	

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regione	Hinweise
Koordinaten des globalen Positionierungssystems (GPS)	`LatLong`	`coordinate` , `coordinates` , `lat long`, `latitude longitude` , `location`, `position`	Any	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 unterstüt

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regione	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 Regione	Hinweise
Vollständiger Name	`Name`	None	Any	CloudWatch In Protokollen können nur vollständige Namen erkannt werden. Unterstützt werden nur lateinische Zeichensätze.

Datentyp	Datenkennungs-ID	Schlüsselwort erforderlich	Länder und Regione	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 numeris`	Any	CloudWatch Protokolle können VINs erkennen, die aus einer Sequenz von 17 Zeichen bestehen und den Normen ISO 3779 und 3780 entsprechen. Diese Standards wurden für den weltweiten Einsatz konzipiert.

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 osterreich
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 prúkaz, 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, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
Griechenland	δεια οδήγησης, adeia odigisis
Ungarn	illesztőprogramok lic, jogosítvány, jogsi, licencszám, 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, führerschäin
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 conducer, el carnet de conducer, licencia conducer, licencia de manejo, número carnet conducer, número de carnet de conducer, número de permiso conducer, número de permiso de conducer, número licencia conducer, número permiso conducer, permiso conducción, permiso conducer, 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’identit é, 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 fiscal, 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	passeport, passeport#, passport, passport#, passportno, passportno#
Frankreich	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non
Deutschland	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass–nr, reisepassnummer
Italien	italian passport number, numéro passeport , numéro passeport 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	passeport #, passeport n °, passeportNon, passeport non, passeportn °, 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 fisica, cadastro de pessoas físicas, cadastro de pessoas fisicas, cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj, cpf
Frankreich	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#
Deutschland	identifikationsnummer, steuer id, steueride ntifikationsnummer, 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)

Datenkennungs-ARNS für persönlich identifizierbare Informationen (PII)

In der folgenden Tabelle werden die Amazon-Ressourcennamen (ARNs) für die persönlich identifizierbaren Informationen (PII) aufgelistet, die Sie Ihren Datenschutzrichtlinien hinzufügen können.

ARNS für PII-Datenkennungen
`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`

ARNS für PII-Datenkennungen
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

ARNS für PII-Datenkennungen
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

ARNS für PII-Datenkennungen
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?

Mit benutzerdefinierten Datenkennungen (Custom data identifiers; 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 Datenkennung verwenden, um nach unternehmensspezifischen Mitarbeiter-IDs zu suchen. Benutzerdefinierte Datenkennungen können in Verbindung mit verwalteten Datenkennungen verwendet werden.

Einschränkungen für benutzerdefinierte Datenkennungen

CloudWatch Für benutzerdefinierte Datenbezeichner 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 Datenkennungen 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-IDs enthalten.

1. Erstellen Sie einen `Configuration`-Block innerhalb Ihrer Datenschutzrichtlinie.
2. Geben Sie eine `Name` als benutzerdefinierte Datenkennung ein. z. B. **`EmployeeId`**.
3. Geben Sie eine `Regex` als benutzerdefinierte Datenkennung ein. z. B. **`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.

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.

> **Note**
>
> Metrikfilter werden nur für Protokollgruppen der Standard-Protokollklasse unterstützt. Weitere Hinweise zu Protokollklassen finden Sie unterKlassen protokollieren.

Sie können jede Art von CloudWatch Statistik verwenden, einschließlich Perzentilstatistiken, wenn Sie diese Metriken anzeigen 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. Gefilterte Ergebnisse geben die ersten 50 Zeilen zurück, die nicht angezeigt werden, wenn der Zeitstempel für die gefilterten Ergebnisse vor der Metrikerstellung liegt.

Inhalt

Konzepte

Jeder Metrikfilter besteht aus den folgenden Schlüsselelementen:

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 ein Teil der eindeutigen ID für eine Metrik sind, erstellen Sie eine neue Variation dieser Metrik, sobald ein eindeutiges Namen-Werte-Paar aus Ihren Protokollen extrahiert wird.

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 unterCloudWatch Syntax der Logs Insights-Abfrage.
> 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 MetricDatumin der Amazon CloudWatch API-Referenz.

Themen

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/Wert-Paare 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 versehentlich hohe Kosten anfallen, kann Amazon Ihren Metrikfilter deaktivieren, wenn er innerhalb eines bestimmten Zeitraums 1 000 verschiedene Name/Wert-Paare für bestimmte Dimensionen erzeugt.

Sie können Rechnungsalarme erstellen, die Sie über Ihre geschätzten Gebühren informieren. Weitere Informationen finden Sie unter Einen Fakturierungsalarm erstellen, um Ihre geschätzten AWS Gebühren zu überwachen.

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) 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.
7. Wählen Sie Next (Weiter) aus und geben Sie einen Namen für Ihren Metrikfilter ein.
8. 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.
9. Unter Metric name (Metrikname) geben Sie einen Namen für die neue Metrik ein.
10. 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.

11. (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.
12. (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.

13. 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 Metrikfilter mit dem zu erstellen 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 MyNameSpaceSie unter Metric Details für Metric Namespace den Text 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 Metrikfilter mit dem zu erstellen 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 für Filtername HTTP404Errors ein.
7. Geben Sie unter Metric Details (Metrikdetails) für Metric Namespace (Metrik-Namespace) **`MyNameSpace`** ein.
8. Geben Sie für Metrikname **`ApacheNotFoundErrorCount`** 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 mit dem zu erstellen 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=ApacheNotFoundErrorCount,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": "ApacheNotFoundErrorCount"
                }
            ],
            "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 für Metric name (Metrikname) HTTP4xxErrors 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 mit dem zu erstellen 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,dimensio
$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 Metrikfilter 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 Ausgabebeispiel:

```
{
    "metricFilters": [
        {
            "filterName": "HTTP404Errors",
            "metricTransformations": [
                {
                    "metricValue": "1",
                    "metricNamespace": "MyNamespace",
                    "metricName": "ApacheNotFoundErrorCount"
                }
            ],
            "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.
5. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Delete (Löschen).

Um einen Metrikfilter mit dem 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.

Wenn Sie Protokollereignisse abonnieren möchten, erstellen Sie die empfangende Ressource (z. B. einen Kinesis-Data-Streams-Datenstrom), an die die Ereignisse gesendet werden sollen. 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.

Sie können Abonnements auf Konto- und Protokollgruppenebene erstellen. Jedes Konto kann über einen Abonnementfilter auf Kontoebene verfügen. Jeder Protokollgruppe können bis zu zwei Abonnementfilter zugeordnet sein.

> **Note**
>
> Wenn der Zieldienst einen Fehler zurückgibt, der wiederholt werden kann, z. B. eine Drosselungsausnahme oder eine Serviceausnahme, 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 unterKlassen 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.

Inhalt

Konzepte

Jeder Abonnementfilter besteht aus den folgenden Schlüsselelementen:

Filtermuster

Eine symbolische Beschreibung, wie CloudWatch Logs die Daten in den einzelnen Protokollereignissen interpretieren sollte, zusammen mit Filterausdrücken, die einschränken, was an die AWS Zielressource übermittelt 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 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 Schlüsselelement 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 Schlüsselelement 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 unterVerhinderung der Rekursion von Protokollen.

Für die Auswahlkriterien gilt eine Größenbeschränkung von 25 KB.

Abonnementfilter auf Gruppenebene protokollieren

Sie können einen Abonnementfilter mit Kinesis Data Streams, Lambda oder Firehose verwenden. Alle Protokolle, die über einen Abonnementfilter an einen empfangenden Service gesendet werden, sind base64-kodiert und im GZIP-Format komprimiert.

Sie können Ihre Protokolldaten mithilfe der Filter- und Mustersyntax suchen.

Beispiele

Beispiel 1: Abonnementfilter mit Kinesis Data Streams

Im folgenden Beispiel wird ein Abonnementfilter einer Protokollgruppe zugeordnet, die Ereignisse enthält. AWS CloudTrail Der Abonnementfilter überträgt jede protokollierte Aktivität, die mit „Root“ AWS -Anmeldeinformationen ausgeführt wird, an einen Stream in 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:
>
> - Überwachen Sie Ihren Stream mithilfe von CloudWatch Metriken. 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 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 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 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 Kinesis Data Streams Streams-Befehl describe-stream verwenden, um das zu überprüfen. StreamDescription StreamStatusEigenschaft. 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 Ausgabebeispiel:

```
{
    "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"
                }
            }
        ]
    }
```

```
}
```

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 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": "AAOIIAH450GAB4HC5F431",
        "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. Ordnen Sie die Berechtigungsrichtlinie mithilfe des folgenden put-role-policyBefehls der Rolle zu:

```
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 auch geschieht, indem Sie einen Kinesis-Data-Streams-Shard-Iterator aufrufen und mit dem Kinesis-Data-Streams-Befehl „get-records“ einige Kinesis-Data-Streams-Datensätze abrufen:

```
aws kinesis get-shard-iterator --stream-name RootAccess --shard-id
 shardId-000000000000 --shard-iterator-type TRIM_HORIZON
```

```
{
    "ShardIterator":
    "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+OIK8zM5My8ID
+g6rMo7UKWeI4+IWiK2OSh0uP"
}
```

```
aws kinesis get-records --limit 10 --shard-iterator "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+OIK8zM5My8ID
+g6rMo7UKWeI4+IWiK2OSh0uP"
```

Beachten Sie, dass Sie diesen Befehl möglicherweise mehrmals aufrufen müssen, bevor Kinesis Data Streams Daten zurückgibt.

Sie sollten davon ausgehen, dass die Antwort in einem Datensatz-Array angezeigt wird. Das Attribut Data in einem Kinesis-Data-Streams-Datensatz ist base64-kodiert und im 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\":
\"Root\"}"
        }
    ]
}
```

Die Schlüsselelemente 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 geben CloudWatch Logs Kinesis Data Streams Streams-Datensätze mit dem Typ „CONTROL_MESSAGE“ aus, 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.

   Wenn Sie die Ausgabe der Lambda-Funktion anzeigen möchten, navigieren Sie zu der Lambda-Funktion. Dort wird die Ausgabe unter /aws/lambda/helloworld angezeigt:

   ```
   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\":
\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221569",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221570",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        }
    ]
}
```

Die Schlüsselelemente 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 Mithilfe CloudWatch von Protokollen in Kinesis Data Firehose schreiben.

> **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 my-bucket --create-bucket-configuration
    LocationConstraint=region
   ```

   Das Folgende ist Ausgabebeispiel:

```
{
    "Location": "/my-bucket"
}
```

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": "AAOIIAH450GAB4HC5F431",
```

```
        "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:::my-bucket",
          "arn:aws:s3:::my-bucket/*" ]
    }
  ]
}
```

5. Ordnen Sie die Berechtigungsrichtlinie mithilfe des folgenden put-role-policy Befehls der Rolle zu:

```
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 Rollen- und Bucket-ARNs, die Sie erstellt haben:

```
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:::my-bucket"}'
```

Beachten Sie, dass Firehose automatisch ein Präfix im UTC-Zeitformat YYYY/MM/DD/HH 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-streamBefehl Firehose verwenden, um das DeliveryStreamDescription zu überprüfen. DeliveryStreamStatusEigentum. 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:::my-bucket",
                    "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. Ordnen Sie die Berechtigungsrichtlinie mit dem put-role-policy folgenden Befehl der Rolle zu:

```
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 Protokolldaten in Echtzeit 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 'my-bucket' --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 'my-bucket' --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
```

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 Kinesis Data Streams, Lambda oder Firehose funktionieren. Protokolle, die über eine Abonnementrichtlinie auf Kontoebene an einen Empfangsdienst gesendet werden, sind Base64-kodiert und im GZIP-Format komprimiert.

> **Note**
>
> Um eine Liste aller Abonnementfilterrichtlinien in Ihrem Konto anzuzeigen, verwenden Sie den `describe-account-policies` Befehl mit dem Wert von für den Parameter. `SUBSCRIPTION_FILTER_POLICY` `--policy-type` Weitere Informationen finden Sie unter describe-account-policies¶.

Beispiele

- Beispiel 1: Abonnementfilter mit Kinesis Data Streams
- Beispiel 2: Abonnementfilter mit AWS Lambda
- Beispiel 3: Abonnementfilter mit Amazon Data Firehose

Beispiel 1: Abonnementfilter mit Kinesis Data Streams

Bevor Sie einen 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 Grenzwerte in der 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 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 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 Kinesis Data Streams entspricht. Wenn Sie zu viele Daten an den Stream senden, müssen Sie möglicherweise die Filtergröße reduzieren oder die Filterkriterien anpassen.

Im folgenden Beispiel wird eine Abonnementrichtlinie auf Kontoebene verwendet, um alle Protokollereignisse an einen Stream in Kinesis Data Streams weiterzuleiten. Das Filtermuster ordnet alle Protokollereignisse dem Text zu `Test` und leitet sie an den Stream in Kinesis Data Streams weiter.

So erstellen Sie eine Abonnementrichtlinie auf Kontoebene für 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 Ausgabebeispiel:

```
{
    "StreamDescription": {
        "StreamStatus": "ACTIVE",
        "StreamName": "TestStream",
        "StreamARN": "arn:aws:kinesis:region:123456789012:stream/TestStream",
        "Shards": [
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "EndingHashKey": "EXAMPLE84634633746074317682114555",
                    "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 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": "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. Ordnen Sie die Berechtigungsrichtlinie mithilfe des folgenden put-role-policyBefehls der Rolle zu:

```
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 unterVerhinderung 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 Kinesis Data Streams-Shard-Iterator verwenden und den `get-records` Befehl Kinesis Data Streams verwenden, um einige 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/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+OIK8zM5My8ID
+g6rMo7UKWeI4+IWiK2OSh0uP"
```

```
}
```

```
aws kinesis get-records --limit 10 --shard-iterator "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+OIK8zM5My8ID
+g6rMo7UKWeI4+IWiK2OSh0uP"
```

Möglicherweise müssen Sie diesen Befehl einige Male verwenden, bevor Kinesis Data Streams Daten zurückgibt.

Sie sollten davon ausgehen, dass die Antwort in einem Datensatz-Array angezeigt wird. Das Attribut Data in einem Kinesis-Data-Streams-Datensatz ist base64-kodiert und im 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\":
\"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"
}
```

Die wichtigsten Elemente der Datenstruktur sind die folgenden:

messageType

Datennachrichten verwenden den Typ "DATA_MESSAGE". Manchmal geben CloudWatch Logs 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 unterVerhinderung 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.

   Wenn Sie die Ausgabe der Lambda-Funktion anzeigen möchten, navigieren Sie zu der Lambda-Funktion. Dort wird die Ausgabe unter /aws/lambda/helloworld angezeigt:

   ```
   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 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 my-bucket --create-bucket-configuration
    LocationConstraint=region
   ```

   Das Folgende ist Ausgabebeispiel:

   ```
   {
       "Location": "/my-bucket"
   }
   ```

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:::my-bucket",
          "arn:aws:s3:::my-bucket/*" ]
    }
  ]
}
```

5. Ordnen Sie die Berechtigungsrichtlinie mithilfe des folgenden put-role-policy Befehls der Rolle zu:

```
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 Rollen- und Bucket-ARNs, die Sie erstellt haben:

```
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:::my-bucket"}'
```

NFirehose verwendet automatisch ein Präfix im UTC-Zeitformat YYYY/MM/DD/HH 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-streamBefehl Firehose verwenden, um das DeliveryStreamDescription zu überprüfen. DeliveryStreamStatusEigentum. 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:::my-bucket",
                    "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": "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 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. Ordnen Sie die Berechtigungsrichtlinie mit dem put-role-policy folgenden Befehl der Rolle zu:

```
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 unterVerhinderung 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 'my-bucket' --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 'my-bucket' --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 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. Dabei kann ein Echtzeit-Ereignisdatenstrom für alle diese Konten zusammengestellt und an die Informationssicherheitsgruppen gesendet werden, die die Daten wiederum mithilfe von Kinesis Data Streams mit ihren bereits vorhandenen Sicherheitsanalysesystemen verknüpfen können.

> **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

- Kontoübergreifende, regionsübergreifende Abonnements auf Kontoebene mit Kinesis Data Streams
- Kontoübergreifende, regionsübergreifende Abonnements auf Kontoebene mit Firehose

Kontoübergreifender regionsübergreifender Austausch von Protokolldaten mit 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 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 Schlüsselelemente:

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

Befolgen Sie die Schritte in diesen Abschnitten, um ein neues kontoübergreifendes Protokollabonnement einzurichten.

Themen

- 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 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 Zieldatenstrom in 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": "340282366920938463463374607431768EXAMPLE",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49551135218688818456679503831981458784591352702 18EXAMPLE"
        }
      }
    ]
  }
}
```

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 ~/ TrustPolicyFor CWL.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": "AAOIIAH450GAB4HC5F431",
        "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 ~/ PermissionsFor CWL.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 DestinationArndas, was in der Payload zurückgegeben wird:

```
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 dem Absenderkonto die Erlaubnis erteilen, 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.

```
{
    "Version" : "2012-10-17",
    "Statement" : [
        {
            "Sid" : "",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : "logs:PutSubscriptionFilter",
            "Resource" :
 "arn:aws:logs:region:999999999999:destination:testDestination",
            "Condition": {
               "StringEquals" : {
                   "aws:PrincipalOrgID" : ["o-1234567890"]
                }
            }
        }
    ]
}
```

In diesem nächsten Beispiel kann nur das Protokolldatenabsenderkonto (111111111111) Protokolle an das Protokolldatenempfängerkonto senden.

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
```

```
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111111111111"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" :
 "arn:aws:logs:region: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 der ARN arn:aws:logs: region:999999999999:destination:testDestination anzurufen.* 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 Richtlinienvalidierung im IAM-Benutzerhandbuch.

Wenn Sie fertig sind und AWS Organizations für Ihre kontoübergreifenden Berechtigungen verwenden, folgen Sie den Schritten unterSchritt 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 111111111111 befindet, Berechtigungen erteilt, anstatt Konto 111111111111 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 111111111111.

So erstellen Sie die IAM-Rolle, die für kontoübergreifende Protokollabonnements mit AWS Organizations erforderlich ist

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: Hinzufügen/Überprüfen der 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 AWS -Protokollbefehlen verwendet wird.

```
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 oder dargestellt wird. *RoleNameUserName*

2. Melden Sie sich AWS Management Console 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. In den folgenden Beispielen für Richtlinien werden empfohlene Berechtigungen erläutert.

   Die folgende Richtlinie gewährt Berechtigungen zum Erstellen eines Abonnementfilters für jede Zielressource nur in einem einzigen AWS Konto, einem Konto: `123456789012`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow subscription filters on any resource in one specific
 account",
            "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`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow subscription filters on one specific resource in one
 specific account",
            "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 CloudTrailEreignisse 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: Hinzufügen/Überprüfen der 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 Kinesis Data Streams 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 Zielbesitzer kann überprüfen, ob dies geschieht, indem er den get-shard-iterator Befehl aws kinesis verwendet, um einen Kinesis Data Streams Streams-Shard abzurufen, und den Befehl aws kinesis get-records verwendet, um einige 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/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
```

```
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+OIK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
}

aws kinesis get-records \
      --limit 10 \
      --shard-iterator
      "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+OIK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
```

> **Note**
>
> Sie müssen den Befehl „get-records“ möglicherweise einige Male erneut ausführen, bevor Kinesis Data Streams Daten zurückgibt.

Sie sollten eine Antwort mit einem Array von Kinesis-Data-Streams-Datensätzen erhalten. Das Datenattribut im Kinesis-Data-Streams-Datensatz wird im GZIP-Format komprimiert und dann base64-kodiert. 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": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
```

```
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "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 geben CloudWatch Logs Kinesis Data Streams Streams-Datensätze mit dem Typ „CONTROL_MESSAGE“ aus, 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": "{\"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:

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "222222222222"
```

```
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:region:999999999999:destination:testDestination"
    }
  ]
}
```

3. Rufen Sie PutDestinationPolicyauf, 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

> **Note**
>
> Dieser Schritt ist nur für kontenübergreifende Abonnements für Protokolle erforderlich, die von den in Aktivieren der Protokollierung von AWS Diensten 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"
```

So aktualisieren Sie einen Abonnementfilter, um mit der Verwendung von Organisations-IDs für kontoübergreifende Protokollberechtigungen zu beginnen

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.

```
{
    "Version" : "2012-10-17",
    "Statement" : [
        {
            "Sid" : "",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : "logs:PutSubscriptionFilter",
            "Resource" :
 "arn:aws:logs:region: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 für die Verwendung einer Zugriffsrichtlinie mit einer

Organisations-ID anstelle einer Zugriffsrichtlinie zu aktualisieren, die bestimmte AWS -Konto-IDs auflistet, fügen Sie den `force`-Parameter hinzu.

> **Warning**
>
> Wenn Sie mit Protokollen arbeiten, die von einem AWS Dienst gesendet wurden, der unter aufgeführt istAktivieren der Protokollierung von AWS Diensten, müssen Sie, bevor Sie diesen Schritt ausführen, zuerst die Abonnementfilter in allen Absenderkonten aktualisiert haben, wie unter beschriebenSchritt 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 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

> **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 firehose-test-bucket1 --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": "AROAR3BXASEKW7K635M53",
           "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:::firehose-test-bucket1",
            "arn:aws:s3:::firehose-test-bucket1/*"
        ]
    }]
}
```

   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 *my-bucket-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:::firehose-test-bucket1"}'
```

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:::firehose-test-bucket1",
                    "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:::firehose-test-bucket1",
                    "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": "AROAR3BXASEKYJYWF243H",
        "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:*"
                        ]
                    }
                }
            }
        }
    }
}
```

4. 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:*"]
```

```
      }
    ]
}
```

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 Datei ~/ AccessPolicy .json einzufügen:

```
{
  "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 PutSubscriptionFilterAktion, um Protokollereignisse an das Ziel zu senden:

```
aws logs put-destination-policy \
    --destination-name "testFirehoseDestination" \
    --access-policy file://~/AccessPolicy.json
```

Schritt 3: Hinzufügen/Überprüfen der 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 AWS -Protokollbefehlen verwendet wird.

   ```
   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 oder dargestellt wird. *RoleNameUserName*
2. Melden Sie sich AWS Management Console 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. In den folgenden Beispielen für Richtlinien werden empfohlene Berechtigungen erläutert.

   Die folgende Richtlinie gewährt Berechtigungen zum Erstellen eines Abonnementfilters für jede Zielressource nur in einem einzigen AWS Konto, einem Konto: `123456789012`

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "Allow subscription filters on any resource in one specific
    account",
               "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`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow subscription filters on one specific resource in one
 specific account",
            "Effect": "Allow",
            "Action": "logs:PutSubscriptionFilter",
            "Resource": [
                "arn:aws:logs:*:*:log-group:*",
                "arn:aws:logs:*:123456789012:destination:sampleDestination"
            ]
        }
    ]
}
```

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: Hinzufügen/Überprüfen der 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 'firehose-test-bucket1'
```

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 'firehose-test-bucket1' --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}\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:

```
{
  "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 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 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 Schlüsselelemente:

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

Befolgen Sie die Schritte in diesen Abschnitten, um ein neues kontoübergreifendes Protokollabonnement einzurichten.

Themen

- 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 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 Zieldatenstrom in 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": "340282366920938463463374607431768EXAMPLE",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49551135218688818456679503831981458784591352702 18EXAMPLE"
        }
      }
    ]
  }
}
```

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 ~/ TrustPolicyFor CWL.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": "AAOIIAH450GAB4HC5F431",
        "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 ~/ PermissionsFor CWL.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 DestinationArndas, was in der Payload zurückgegeben wird:

```
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 dem Absenderkonto die Erlaubnis erteilen, 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.

```
{
    "Version" : "2012-10-17",
    "Statement" : [
        {
            "Sid" : "",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
            "Resource" :
 "arn:aws:logs:region:999999999999:destination:testDestination",
            "Condition": {
               "StringEquals" : {
                   "aws:PrincipalOrgID" : ["o-1234567890"]
                }
            }
        }
    ]
}
```

   In diesem nächsten Beispiel kann nur das Protokolldatenabsenderkonto (111111111111) Protokolle an das Protokolldatenempfängerkonto senden.

```
{
```

```
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111111111111"
      },
      "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
      "Resource" :
 "arn:aws:logs:region: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 der ARN arn:aws:logs: region:999999999999:destination:testDestination anzurufen.* 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 Richtlinienvalidierung im IAM-Benutzerhandbuch.

Wenn Sie fertig sind und AWS Organizations für Ihre kontoübergreifenden Berechtigungen verwenden, folgen Sie den Schritten unterSchritt 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 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 111111111111.

So erstellen Sie die IAM-Rolle, die für kontoübergreifende Protokollabonnements mit AWS Organizations erforderlich ist

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 unterVerhinderung 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 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 Zielbesitzer kann überprüfen, ob dies geschieht, indem er den get-shard-iterator Befehl aws kinesis verwendet, um einen Kinesis Data Streams Streams-Shard abzurufen, und den Befehl aws kinesis get-records verwendet, um einige 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/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+OIK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
}

aws kinesis get-records \
      --limit 10 \
      --shard-iterator
      "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+OIK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
```

> **Note**
>
> Möglicherweise müssen Sie den `get-records` Befehl einige Male erneut ausführen, bevor Kinesis Data Streams Daten zurückgibt.

Sie sollten eine Antwort mit einem Array von Kinesis-Data-Streams-Datensätzen erhalten. Das Datenattribut im Kinesis-Data-Streams-Datensatz wird im GZIP-Format komprimiert und dann base64-kodiert. 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": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "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 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:

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "222222222222"
      },
      "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
      "Resource" : "arn:aws:logs:region:999999999999:destination:testDestination"
    }
  ]
}
```

3. Rufen Sie PutDestinationPolicyauf, 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 Aktivieren der Protokollierung von AWS Diensten 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"
```

So aktualisieren Sie einen Abonnementfilter, um mit der Verwendung von Organisations-IDs für kontoübergreifende Protokollberechtigungen zu beginnen

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.

```
{
```

```
"Version" : "2012-10-17",
    "Statement" : [
        {
"Sid" : "",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
            "Resource" :
 "arn:aws:logs:region: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 für die Verwendung einer Zugriffsrichtlinie mit einer Organisations-ID anstelle einer Zugriffsrichtlinie zu aktualisieren, die bestimmte AWS -Konto-IDs auflistet, fügen Sie den `force`-Parameter hinzu.

> **Warning**
>
> Wenn Sie mit Protokollen arbeiten, die von einem der unter aufgeführten AWS Dienste gesendet wurdenAktivieren der Protokollierung von AWS Diensten, müssen Sie, bevor Sie diesen Schritt ausführen, zunächst die Abonnementfilter in allen Absenderkonten aktualisiert haben, wie unter beschriebenSchritt 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 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

- 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 firehose-test-bucket1 --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": "AROAR3BXASEKW7K635M53",
        "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:::firehose-test-bucket1",
            "arn:aws:s3:::firehose-test-bucket1/*"
        ]
    }]
}
```

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 *my-bucket-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:::firehose-test-bucket1"}'
```

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:::firehose-test-bucket1",
                       "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:::firehose-test-bucket1",
                    "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": "AROAR3BXASEKYJYWF243H",
        "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:*"
                        ]
                    }
                }
            }
        }
    }
}
```

4. 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:*"]
      }
    ]
}
```

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:

```
{
  "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 'firehose-test-bucket1'
```

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 'firehose-test-bucket1' --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 PutDestinationPolicyund 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      \"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}\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:

```
{
  "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 privilegiertere Entität zur Durchführung der Aktion zwingen kann. In kann AWS ein dienstübergreifender 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, die Kontextschlüssel `aws:SourceArn` oder die `aws:SourceAccount` globalen Bedingungsschlüssel in Ressourcenrichtlinien zu verwenden, um den Umfang der Berechtigungen einzuschränken, die Sie CloudWatch Logs zum Schreiben von Daten in Kinesis Data Streams und Firehose gewähren.

Der Wert von `aws:SourceArn` muss die Berechtigungen auf die Konten beschränken, die Daten schreiben und empfangen.

Der effektivste Weg, um sich vor dem Confused-Deputy-Problem zu schützen, ist die Verwendung des globalen Bedingungskontextschlüssels `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 Kontext-Bedingungsschlüssel aws: SourceArn global mit Platzhaltern (*) für die unbekannten Teile des ARN. z. B. `arn:aws:servicename::123456789012:*`.

Die dokumentierten Richtlinien zur Gewährung des Zugriffs auf CloudWatch Logs zum Schreiben von Daten in Schritt 1: Erstellen eines Ziels Kinesis Data Streams und Firehose Schritt 2: Erstellen eines Ziels zeigen, wie Sie den Kontextschlüssel aws: SourceArn global condition verwenden können, um das Problem des verwirrten Stellvertreters zu vermeiden.

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 wird in einer Endlosschleife geschehen, 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 unterCloudWatch Syntax der Logs Insights-Abfrage.

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, Filterprotokollereignisse und Live Tail verwenden, um Begriffe in Protokollereignissen zu finden. 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 Konsole testen. CloudWatch

Themen

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 mehr Instanzen des vorherigen 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. `%gra|ey%` stimmt beispielsweise mit „gray“ oder „grey“ überein.

  > **Note**
  >
  > Ein Begriff besteht aus einem einzelnen Zeichen oder einer sich wiederholenden Zeichenklasse, die einen der folgenden Operatoren verwendet: `?`, `*`, `+` oder `{n,m}`.

- `\`: 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 hexadezimales 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 mit anderen Filtermustern wie dem Ein- und Ausschließen von Begriffen kombinieren. Wenn Sie „?“ mit anderen Filtermustern kombinieren, wird das Fragezeichen („?“) ignoriert.
> Das folgende Filtermuster entspricht beispielsweise allen Ereignissen, die das Wort `REQUEST` enthalten, aber der Fragezeichenfilter („?“) wird ignoriert und hat 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 Beispiel für ein Filtermuster, 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 Beispiel für ein Filtermuster, 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 einen 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]
```

Aktivieren der Protokollierung von AWS Diensten

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 Protokolle direkt in Amazon S3 oder Firehose veröffentlicht werden, fallen Gebühren an. Weitere Informationen finden Sie unter Verkaufte Logs auf der Registerkarte Logs bei Amazon CloudWatch Pricing.

Einige AWS Dienste verwenden eine gemeinsame Infrastruktur, um ihre Protokolle zu senden. 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 Services, die diese zusätzlichen Berechtigungen erfordern, sind in der Tabelle als Unterstützte [V1 Berechtigungen] und Unterstützte [V2 Berechtigungen] aufgeführt. Informationen zu diesen erforderlichen Berechtigungen finden Sie in den Abschnitten nach der Tabelle.

Protokolltyp	CloudWatch Logs	Amazon S3	Firehose
Zugriffsprotokolle für Amazon API Gateway	Unterstützte [V1 Berechtigungen]		
AWS AppSync logs	Unterstützt		
Amazon-Aurora-MySQL-Protokolle	Unterstützt		
Amazon-Chime-Metrik-Protokolle zur Medienqualität und SIP-Nachrichtenprotokolle	Unterstützte [V1 Berechtigungen]		

Protokolltyp	CloudWatch Logs	Amazon S3	Firehose
CloudFront: auf Protokolle zugreifen		Unterstützte [V1 Berechtigungen]	
AWS CloudHSM Audit-Protokolle	Unterstützt		
CloudWatch Offensichtlich Evaluierungsereignisprotokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	
CloudWatch Internet Monitor-Protokolle		Unterstützte [V1 Berechtigungen]	
CloudTrail Logs	Unterstützt		
AWS CodeBuild logs	Unterstützt		
Amazon CodeWhisperer Ereignisprotokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]
Amazon Cognito logs	Unterstützte [V1 Berechtigungen]		
Amazon-Connect-Protokolle	Unterstützt		
AWS DataSync logs	Unterstützt		
Amazon ElastiCache für Redis-Protokolle	Unterstützte [V1 Berechtigungen]		Unterstützte [V1 Berechtigungen]
AWS Elastic Beanstalk logs	Unterstützt		

Protokolltyp	CloudWatch Logs	Amazon S3	Firehose
Protokolle des Amazon Elastic Container Service	Unterstützt		
Protokolle der Steuerungsebene des Amazon Elastic Kubernetes Service	Unterstützt		
AWS Fargate logs	Unterstützt		
AWS Fault Injection Service Versuchsprotokolle		Unterstützte [V1 Berechtigungen]	
Amazon FinSpace	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]
AWS Global Accelerator Flow-Protokolle		Unterstützte [V1 Berechtigungen]	
AWS Glue Auftragsprotokolle	Unterstützt		
IAM Identity Center-Fehlerprotokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]
Chat-Protokolle von Amazon Interactive Video Service	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]
AWS IoT logs	Unterstützt		
AWS IoT FleetWise logs	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]

Protokolltyp	CloudWatch Logs	Amazon S3	Firehose
AWS Lambda logs	Unterstützt		
Amazon-Macie-Protokolle	Unterstützt		
AWS Mainframe Modernization	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]
Protokolle von Amazon Managed Service für Prometheus	Unterstützte [V1 Berechtigungen]		
Amazon-MSK-Broker-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]
Amazon-MSK-Connect-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]
Allgemeine und Prüfungsprotokolle von Amazon MQ	Unterstützt		
AWS Netzwerk-Firewall-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]
Network-Load-Balancer-Zugriffsprotokolle		Unterstützte [V1 Berechtigungen]	
OpenSearch Logs	Unterstützt		
Amazon OpenSearch Service-Aufnahmeprotokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]

Protokolltyp	CloudWatch Logs	Amazon S3	Firehose
AWS OpsWorks logs	Unterstützt		
ServicePostgreSQL-Protokolle für Amazon Relational Database	Unterstützt		
AWS RoboMaker Logs	Unterstützt		
Öffentliche DNS-Abfrageprotokolle von Amazon Route 53	Unterstützt		
Amazon-Route-53-Resolver-Abfrageprotokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	
SageMaker Amazon-Veranstaltungen	Unterstützte [V1 Berechtigungen]		
Veranstaltungen Amazon SageMaker Amazon-Mitarbeiter	Unterstützte [V1 Berechtigungen]		
AWS Site-to_Site-VPN-Protokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]
Protokolle des Amazon Simple Notification Service	Unterstützt		
Datenschutzrichtlinienprotokolle des Amazon Simple Notification Service	Unterstützt		
EC2-Spot-Instance-Daten-Feed-Dateien		Unterstützte [V1 Berechtigungen]	

Protokolltyp	CloudWatch Logs	Amazon S3	Firehose
AWS Step Functions Express-Workflow- und Standard-Workflow-Protokolle	Unterstützte [V1 Berechtigungen]		
Storage-Gateway-Überwachungsprotokolle und -Zustandsprotokolle	Unterstützte [V1 Berechtigungen]		
AWS Transfer Family logs	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]
AWS Verified Access logs	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]
Amazon-Virtual-Private-Cloud-Flow-Protokolle	Unterstützt	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]
Amazon-VPC-Lattice-Zugriffsprotokolle	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]
AWS WAF logs	Unterstützte [V1 Berechtigungen]	Unterstützte [V1 Berechtigungen]	Unterstützt
Amazon WorkMail Protokolle	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]	Unterstützte [V2 Berechtigungen]

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. So aktivieren Sie die AWS -Services, die in der folgenden Tabelle aufgeführt sind, um ihre Protokolle an diese Ziele zu senden. Sie müssen als Benutzer angemeldet sein, der über bestimmte Berechtigungen verfügt.

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 der folgenden Tabelle wird zusammengefasst, welche Arten von Protokollen und welche Protokollziele die Informationen in diesem Abschnitt betreffen.

In den folgenden Abschnitten erhalten Sie weitere Details zu diesen Zielen.

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**
>
> Achten Sie bei der Angabe der `logs:DescribeLogGroups` Berechtigung 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:PutResourcePolicylogs: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.

```
{
  "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-1:0123456789:log-group:my-log-group:log-stream:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": ["0123456789"]
        },
        "ArnLike": {
          "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:*"]
        }
      }
    }
  ]
}
```

Wenn die Protokollgruppe über eine Ressourcenrichtlinie verfügt, diese Richtlinie jedoch nicht die in der vorherigen Richtlinie gezeigte Anweisung enthält und der Benutzer, der die Protokollierung einrichtet, die `logs:PutResourcePolicy`-, `logs:DescribeResourcePolicies`-, und`logs:DescribeLogGroups`-Berechtigungen für die Protokollgruppe hat, wird diese Anweisung an die Ressourcenrichtlinie der Protokollgruppe angehängt.

Überlegungen zur Größenbeschränkung der Protokollgruppen-Ressourcenrichtlinie

Diese Dienste müssen jede Protokollgruppe, an die sie Protokolle senden, in der Ressourcenrichtlinie auflisten, und die Ressourcenrichtlinien für CloudWatch Protokolle sind auf 5120 Zeichen begrenzt. Bei einem Dienst, der Protokolle an eine große Anzahl von Protokollgruppen sendet, kann dieses Limit erreicht werden.

Um dieses Problem zu minimieren, überwacht CloudWatch Logs die Größe der Ressourcenrichtlinien, die von dem Dienst verwendet werden, der Protokolle sendet. Wenn festgestellt wird, dass sich eine Richtlinie der Größenbeschränkung von 5120 Zeichen nähert, aktiviert CloudWatch Logs automatisch die Option Logs `/aws/vendedlogs/*` in der Ressourcenrichtlinie für diesen Dienst. Sie können dann anfangen, Protokollgruppen als Ziele für Protokolle aus diesen Services zu verwenden, deren Namen mit `/aws/vendedlogs/` beginnt.

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.

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, über die `S3:GetBucketPolicy` und `S3:PutBucketPolicy` -Berechtigungen für den Bucket verfügt, erstellt er AWS automatisch die folgende Richtlinie dafür, wenn Sie beginnen, die Protokolle an Amazon S3 zu senden.

```
{
    "Version": "2012-10-17",
    "Id": "AWSLogDeliveryWrite20150319",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryAclCheck",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
                },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::my-bucket",
            "Condition": {
                "StringEquals": {
                "aws:SourceAccount": ["0123456789"]
                },
                "ArnLike": {
                "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:*"]
                }
            }
        },
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-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-1:0123456789:*"]
                }
            }
        }
    ]
}
```

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 von Amazon S3 verwalteten Schlüsseln (SSE-S3) oder die serverseitige Verschlüsselung mit einem in (SSE-KMS) gespeicherten AWS KMS Schlüssel aktivieren. AWS Key Management Service Weitere Informationen finden Sie unter Schützen von Daten mithilfe serverseitiger Verschlüsselung.

Wenn Sie SSE-S3 wählen, ist keine zusätzliche Konfiguration erforderlich. Amazon S3 verarbeitet den Verschlüsselungsschlüssel.

> **Warning**
>
> Wenn Sie SSE-KMS wählen, müssen Sie einen vom Kunden verwalteten Schlüssel verwenden, da die Verwendung eines verwalteten Schlüssels in diesem Szenario nicht unterstützt wird. AWS Wenn Sie die Verschlüsselung mit einem AWS verwalteten Schlüssel einrichten, werden die Protokolle in einem unlesbaren Format übermittelt.

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 SSE-KMS wählen, 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 dienstverknüpfte Rolle mit dem Namen. AWSServiceRoleForLogDelivery Diese serviceverknüpfte Rolle umfasst die folgenden Berechtigungen.

```
{
```

```
    "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:

```
{
  "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.

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 PutDeliveryDestinationPolicyim 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 DeleteDeliveryOperation.
2. Löschen Sie die `DeliverySource` mithilfe der DeleteDeliverySourceOperation.
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 DeleteDeliveryDestinationsVorgangs entfernen.

Inhalt

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.

```
{
    "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"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:delivery:*",
                "arn:aws:logs:region:account-id:delivery-source:*",
                "arn:aws:logs:region:account-id:delivery-destination:*"
            ]
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyCWL",
            "Effect": "Allow",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:*"
            ]
        }
    ]
}
```

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:PutResourcePolicylogs: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.

```
{
  "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-1:0123456789:log-group:my-log-group:log-stream:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": ["0123456789"]
        },
        "ArnLike": {
          "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:*"]
        }
      }
    }
  ]
}
```

Überlegungen zur Größenbeschränkung der Protokollgruppen-Ressourcenrichtlinie

Diese Dienste müssen jede Protokollgruppe, an die sie Protokolle senden, in der Ressourcenrichtlinie auflisten, und die Ressourcenrichtlinien für CloudWatch Protokolle sind auf 5120 Zeichen begrenzt.

Ein Service, der Protokolle an eine große Anzahl von Protokollgruppen sendet, kann durchaus diesen Grenzwert erreichen.

Um dies zu vermeiden, überwacht CloudWatch Logs die Größe der Ressourcenrichtlinien, die von dem Dienst verwendet werden, der die Protokolle sendet. Wenn festgestellt wird, dass eine Richtlinie die Größenbeschränkung von 5120 Zeichen erreicht, aktiviert CloudWatch Logs automatisch die Option Logs `/aws/vendedlogs/*` in der Ressourcenrichtlinie für diesen Dienst. Sie können dann anfangen, Protokollgruppen als Ziele für Protokolle aus diesen Services zu verwenden, deren Namen mit `/aws/vendedlogs/` beginnt.

An Amazon S3 gesendete Protokolle

Benutzerberechtigungen

Um das Senden von Protokollen an Amazon S3 zu aktivieren, müssen Sie mit den folgenden Berechtigungen angemeldet sein.

```
{
    "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"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:delivery:*",
                "arn:aws:logs:region:account-id:delivery-source:*",
                "arn:aws:logs:region:account-id:delivery-destination:*"
            ]
```

```
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries"
            ],
            "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, über die `S3:GetBucketPolicy` und `S3:PutBucketPolicy` -Berechtigungen für den Bucket verfügt, erstellt er AWS automatisch die folgende Richtlinie dafür, wenn Sie beginnen, die Protokolle an Amazon S3 zu senden.

```
{
    "Version": "2012-10-17",
    "Id": "AWSLogDeliveryWrite20150319",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryAclCheck",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
                },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::my-bucket",
            "Condition": {
                "StringEquals": {
```

```
                "aws:SourceAccount": ["0123456789"]
                },
                "ArnLike": {
                "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:delivery-source*"]
                }
            }
        },
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-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-1:0123456789: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 von Amazon S3 verwalteten Schlüsseln (SSE-S3) oder die serverseitige Verschlüsselung mit einem in (SSE-KMS) gespeicherten AWS KMS Schlüssel aktivieren. AWS Key Management Service Weitere Informationen finden Sie unter Schützen von Daten mithilfe serverseitiger Verschlüsselung.

Wenn Sie SSE-S3 wählen, ist keine zusätzliche Konfiguration erforderlich. Amazon S3 verarbeitet den Verschlüsselungsschlüssel.

> **Warning**
>
> Wenn Sie SSE-KMS wählen, müssen Sie einen vom Kunden verwalteten Schlüssel verwenden, da die Verwendung eines verwalteten Schlüssels in diesem Szenario nicht unterstützt wird. AWS Wenn Sie die Verschlüsselung mit einem AWS verwalteten Schlüssel einrichten, werden die Protokolle in einem unlesbaren Format übermittelt.

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 SSE-KMS wählen, 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.

```
{
    "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"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:delivery:*",
                "arn:aws:logs:region:account-id:delivery-source:*",
                "arn:aws:logs:region:account-id:delivery-destination:*"
            ]
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyFH",
            "Effect": "Allow",
            "Action": [
```

```
                "firehose:TagDeliveryStream"
            ],
            "Resource": [
                "arn:aws:firehose:region:account-id:deliverystream/*"
            ]
        },
        {
            "Sid": "CreateServiceLinkedRole",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::account-id: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 dienstverknüpfte Rolle mit dem Namen. AWSServiceRoleForLogDelivery Diese serviceverknüpfte Rolle umfasst die folgenden Berechtigungen.

```
{
    "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:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Dienstspezifische 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 und ersetzen Sie *`service`* und *`resource-type`* durch die entsprechenden Werte. Die dienstspezifischen Werte für diese Felder finden Sie auf der Dokumentationsseite dieser Dienste für verkaufte Protokolle.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ServiceLevelAccessForLogDelivery",
            "Effect": "Allow",
            "Action": [
```

```
                "service:AllowVendedLogDeliveryForResource"
            ],
            "Resource": "arn:aws:service:region:account-id:resource-type/*"
        }
    ]
}
```

Konsolenspezifische 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:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowLogDeliveryActionsConsoleCWL",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "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": [
                "*"
            ]
        }
    ]
}
```

Serviceübergreifende Confused-Deputy-Prävention

Das Confused-Deputy-Problem ist ein Sicherheitsproblem, bei dem eine juristische Stelle, die nicht über die Berechtigung zum Ausführen einer Aktion verfügt, eine privilegiertere juristische Stelle zwingen kann, die Aktion auszuführen. In AWS kann ein dienstübergreifendes Identitätswechsels zum Problem des verwirrten Stellvertreters führen. Ein dienstübergreifender Identitätswechsel kann auftreten, wenn ein Dienst (der Anruf-Dienst) einen anderen Dienst anruft (den aufgerufenen Dienst). Der aufrufende Service kann manipuliert werden, um seine Berechtigungen zu verwenden, um Aktionen auf die Ressourcen eines anderen Kunden auszuführen, für die er sonst keine Zugriffsberechtigung haben sollte. Um dies zu verhindern, bietet AWS Tools, mit denen Sie Ihre Daten für alle Services mit Serviceprinzipalen schützen können, die Zugriff auf Ressourcen in Ihrem Konto erhalten haben.

Wir empfehlen, die Kontextschlüssel `aws:SourceArn`und die `aws:SourceAccount`globalen Bedingungsschlüssel in Ressourcenrichtlinien zu verwenden, um die Berechtigungen einzuschränken, die CloudWatch Logs und Amazon S3 den Services gewähren, die Protokolle generieren. Wenn Sie beide globalen Bedingungskontextschlüssel verwenden, müssen der `aws:SourceAccount`-Wert und das Konto im `aws:SourceArn`-Wert dieselbe Konto-ID verwenden, wenn sie in derselben Richtlinienanweisung verwendet werden.

Die Werte von `aws:SourceArn` müssen die ARNs der Bereitstellungsquellen sein, die Protokolle generieren.

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 Platzhaltern (*) für die unbekannten Teile des ARN.

Die Richtlinien in den vorherigen Abschnitten dieser Seite zeigen, wie Sie die globalen Bedingungskontextschlüssel `aws:SourceArn` und `aws:SourceAccount` verwenden können, um das Confused-Deputy-Problem zu verhindern.

CloudWatch Protokolliert Aktualisierungen der AWS verwalteten Richtlinien

Details zu Aktualisierungen der AWS verwalteten Richtlinien für CloudWatch Logs anzeigen, 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.

Änderung	Beschreibung	Datum
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	10. Juni 2021

Änderung	Beschreibung	Datum
	ihm verwalteten Richtlinien nachzuverfolgen.	

Exportieren von Protokolldaten nach Amazon S3

Exportieren Sie Protokolldaten aus den Protokollgruppen in einen Amazon-S3-Bucket und verwenden Sie diese Daten zur benutzerdefinierten Verarbeitung und Analyse oder zum Laden in andere Systeme. Sie können in einen Bucket im selben Konto oder in ein anderes Konto 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

> **Note**
>
> Der Export nach Amazon S3 wird nur für Protokollgruppen der Standard-Protokollklasse unterstützt. Weitere Informationen zu Protokollklassen finden Sie unterKlassen protokollieren.

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 die Protokolldaten für jeden Exportvorgang voneinander zu trennen, können Sie ein Präfix angeben, das als Amazon-S3-Schlüsselpräfix für alle exportierten Objekten verwendet wird.

> **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

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 zu 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`my-exported-logs`.

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 im selben Konto
- Kontenübergreifender Export

Export im selben Konto

Wenn sich der Amazon-S3-Bucket im selben Konto befindet wie die zu exportierenden Protokolle, verwenden Sie die Anweisungen in diesem Abschnitt.

Themen

Schritt 1: Einen Amazon-S3-Bucket erstellen

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 S3-Bucket muss sich in derselben Region befinden wie die zu exportierenden Protokolldaten. CloudWatch Logs unterstützt den Export von Daten in S3-Buckets in einer anderen Region nicht.

So erstellen Sie einen S3-Bucket

1. Öffnen Sie die Amazon-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.

Schritt 2: Einrichten von Zugriffsberechtigungen

Um die Exportaufgabe in Schritt 5 erstellen zu können, müssen Sie mit der IAM-Rolle `AmazonS3ReadOnlyAccess` und den folgenden Berechtigungen angemeldet sein:

- `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 Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer 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.

Schritt 3: Festlegen von Berechtigungen für einen S3-Bucket

Standardmäßig werden alle S3-Buckets und -Objekte als privat eingestuft. Nur der Ressourceneigentümer, das AWS-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.

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 Konto-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-Konsole den Bucket aus, den Sie in Schritt 1 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 `my-exported-logs` in den Namen Ihres S3-Bucket. Achten Sie auf die Angabe des korrekten Regionsendpunkts (beispielsweise `us-west-1`) für Prinzipal.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs",
          "Principal": { "Service": "logs.Region.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "aws:SourceAccount": [
                    "AccountId1",
                    "AccountId2",
                    ...
```

```
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:Region:AccountId1:log-group:*",
                        "arn:aws:logs:Region:AccountId2:log-group:*",
                        ...
                     ]
            }
          }
      },
      {
          "Action": "s3:PutObject" ,
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs/*",
          "Principal": { "Service": "logs.Region.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control",
                "aws:SourceAccount": [
                    "AccountId1",
                    "AccountId2",
                    ...
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:Region:AccountId1:log-group:*",
                        "arn:aws:logs:Region:AccountId2: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 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) Schritt 4: Export in einen mit SSE-KMS verschlüsselten Bucket

Dieser Schritt ist nur erforderlich, wenn Sie in einen 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. Wählen Sie anschließend 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, ersetzen Sie *Region* durch die Region Ihrer Protokolle und *Account-ARN* (Konto-ARN) durch den ARN des Kontos, das den KMS-Schlüssel besitzt.

```
{
    "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-Konsole unter https://console.aws.amazon.com/s3/.
16. Wählen Sie den Bucket, den Sie in Schritt 1: Einen S3-Bucket erstellen 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 auswählen“ 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.

Schritt 5: Erstellen einer Exportaufgabe

In diesem Schritt erstellen Sie die Exportaufgabe zum Exportieren 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 Schritt 2: Einrichten von Zugriffsberechtigungen 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.

Kontenübergreifender Export

Wenn sich der Amazon-S3-Bucket in einem anderen Konto befindet als die zu exportierenden Protokolle, verwenden Sie die Anweisungen in diesem Abschnitt.

Themen

Schritt 1: Einen Amazon-S3-Bucket erstellen

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 befinden wie die zu exportierenden Protokolldaten. CloudWatch Logs unterstützt den Export von Daten in S3-Buckets in einer anderen Region nicht.

So erstellen Sie einen S3-Bucket

1. Öffnen Sie die Amazon-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.

Schritt 2: Einrichten von Zugriffsberechtigungen

Zunächst müssen Sie eine neue IAM-Richtlinie erstellen, damit CloudWatch Logs die `s3:PutObject` Berechtigung für den Amazon S3 S3-Ziel-Bucket im Zielkonto erhält.

Welche Richtlinie Sie erstellen, hängt davon ab, ob der Ziel-Bucket AWS KMS Verschlüsselung verwendet.

So erstellen Sie eine IAM-Richtlinie für den Export von Protokollen in einen Amazon-S3-Bucket

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
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.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-exported-logs/*"
        }
    ]
}
```

Wenn der Ziel-Bucket AWS KMS Verschlüsselung verwendet, fügen Sie die folgende Richtlinie in den Editor ein.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::my-exported-logs/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "ARN_OF_KMS_KEY"
    }
  ]
}
```

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 die Exportaufgabe in Schritt 5 erstellen zu können, müssen Sie mit der IAM-Rolle `AmazonS3ReadOnlyAccess` angemeldet sein. Sie müssen außerdem mit der IAM-Richtlinie angemeldet sein, die Sie gerade erstellt haben, sowie mit den folgenden Berechtigungen:

- `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 Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer 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.

Schritt 3: Festlegen von Berechtigungen für einen S3-Bucket

Standardmäßig werden alle S3-Buckets und -Objekte als privat eingestuft. Nur der Ressourceneigentümer, das AWS-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.

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 Konto-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-Konsole den Bucket aus, den Sie in Schritt 1 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 `my-exported-logs` in den Namen Ihres S3-Bucket. Achten Sie auf die Angabe des korrekten Regionsendpunkts (beispielsweise `us-west-1`) für Prinzipal.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs",
          "Principal": { "Service": "logs.Region.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "aws:SourceAccount": [
                    "AccountId1",
                    "AccountId2",
                    ...
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:Region:AccountId1:log-group:*",
                        "arn:aws:logs:Region:AccountId2:log-group:*",
                        ...
                     ]
            }
```

```
            }
        },
        {
            "Action": "s3:PutObject" ,
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::my-exported-logs/*",
            "Principal": { "Service": "logs.Region.amazonaws.com" },
            "Condition": {
              "StringEquals": {
                  "s3:x-amz-acl": "bucket-owner-full-control",
                  "aws:SourceAccount": [
                      "AccountId1",
                      "AccountId2",
                      ...
                  ]
              },
              "ArnLike": {
                      "aws:SourceArn": [
                          "arn:aws:logs:Region:AccountId1:log-group:*",
                          "arn:aws:logs:Region:AccountId2:log-group:*",
                          ...
                      ]
              }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
              "AWS": "arn:aws:iam::create_export_task_caller_account:role/role_name"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-exported-logs/*",
            "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) Schritt 4: Export in einen mit SSE-KMS verschlüsselten Bucket

Dieser Schritt ist nur erforderlich, wenn Sie in einen 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. Wählen Sie anschließend 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, ersetzen Sie *Region* durch die Region Ihrer Protokolle und *Account-ARN* (Konto-ARN) durch den ARN des Kontos, das den KMS-Schlüssel besitzt.

```
{
    "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": "*"
        },
```

```
        {
            "Sid": "Enable IAM Role Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS":
 "arn:aws:iam::create_export_task_caller_account:role/role_name"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "ARN_OF_KMS_KEY"
        }
    ]
}
```

14. Wählen Sie Änderungen speichern aus.
15. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
16. Wählen Sie den Bucket, den Sie in Schritt 1: Einen S3-Bucket erstellen 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 auswählen“ 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.

Schritt 5: Erstellen einer Exportaufgabe

In diesem Schritt erstellen Sie die Exportaufgabe zum Exportieren 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 Schritt 2: Einrichten von Zugriffsberechtigungen 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 exportieren`my-exported-logs`. 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 im selben Konto

Wenn sich der Amazon-S3-Bucket im selben Konto befindet wie die zu exportierenden Protokolle, verwenden Sie die Anweisungen in diesem Abschnitt.

Themen

Schritt 1: Einen S3-Bucket erstellen

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 befinden wie die zu exportierenden Protokolldaten. 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 my-exported-logs --create-bucket-configuration
 LocationConstraint=us-east-2
```

Es folgt eine Beispielausgabe.

```
{
    "Location": "/my-exported-logs"
}
```

Schritt 2: Einrichten von Zugriffsberechtigungen

Um die Exportaufgabe in Schritt 5 erstellen zu können, müssen Sie mit der IAM-Rolle `AmazonS3ReadOnlyAccess` und den folgenden Berechtigungen angemeldet sein:

- `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 Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer 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.

Schritt 3: Festlegen von Berechtigungen für einen S3-Bucket

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 Konto-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 `my-exported-logs` in den Namen Ihres S3-Bucket und `Principal` in den Endpunkt der Region, in der Sie Protokolldaten exportieren (beispielsweise `us-west-1`). Verwenden Sie einen Text-Editor, um diese Richtliniendatei zu erstellen. Verwenden Sie nicht die IAM-Konsole.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs",
          "Principal": { "Service": "logs.Region.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "aws:SourceAccount": [
                    "AccountId1",
```

```
                    "AccountId2",
                    ...
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:Region:AccountId1:log-group:*",
                        "arn:aws:logs:Region:AccountId2:log-group:*",
                        ...
                     ]
            }
        }
    },
    {
        "Action": "s3:PutObject" ,
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::my-exported-logs/*",
        "Principal": { "Service": "logs.Region.amazonaws.com" },
        "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control",
                "aws:SourceAccount": [
                    "AccountId1",
                    "AccountId2",
                    ...
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:Region:AccountId1:log-group:*",
                        "arn:aws:logs:Region:AccountId2:log-group:*",
                        ...
                    ]
            }
        }
    }
  ]
}
```

2. Legen Sie die Richtlinie, die Sie gerade hinzugefügt haben, mithilfe des put-bucket-policyBefehls 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 my-exported-logs --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) Schritt 4: Export in einen mit SSE-KMS verschlüsselten Bucket

Dieser Schritt ist nur erforderlich, wenn Sie in einen 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. 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:

   - Ersetzen Sie *`Region`* durch die Region Ihrer Protokolle.
   - Ersetzen Sie *`Account-ARN`* (Konto-ARN) durch den ARN des Kontos, das Eigentümer des KMS-Schlüssels ist.

```
{
    "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_id: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
    }
```

3. 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
    }
  ]
}
```

4. Geben Sie den folgenden Befehl ein und ersetzen Sie dabei *`bucket-name`* durch den Namen des Bucket, in den Sie Protokolle exportieren.

```
aws s3api put-bucket-encryption --bucket bucket-name --server-side-encryption-
configuration file://bucketencryption.json
```

   Wenn der Befehl keinen Fehler zurückgibt, ist der Vorgang erfolgreich abgeschlossen.

Schritt 5: Erstellen einer Exportaufgabe

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 Schritt 2: Einrichten von Zugriffsberechtigungen dokumentiert.

2. Verwenden Sie in einer Befehlszeile den folgenden create-export-taskBefehl, 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 "my-exported-logs" --destination-prefix "export-
task-output"
```

Es folgt eine Beispielausgabe.

```
{
    "taskId": "cda45419-90ea-4db5-9833-aade86253e66"
}
```

Kontenübergreifender Export

Wenn sich der Amazon-S3-Bucket in einem anderen Konto befindet als die zu exportierenden Protokolle, verwenden Sie die Anweisungen in diesem Abschnitt.

Themen

- Schritt 1: Einen S3-Bucket erstellen
- Schritt 2: Einrichten von Zugriffsberechtigungen
- Schritt 3: Festlegen von Berechtigungen für einen S3-Bucket
- (Optional) Schritt 4: Export in einen mit SSE-KMS verschlüsselten Bucket
- Schritt 5: Erstellen einer Exportaufgabe

Schritt 1: Einen S3-Bucket erstellen

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 befinden wie die zu exportierenden Protokolldaten. 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 my-exported-logs --create-bucket-configuration
 LocationConstraint=us-east-2
```

Es folgt eine Beispielausgabe.

```
{
    "Location": "/my-exported-logs"
}
```

Schritt 2: Einrichten von Zugriffsberechtigungen

Zunächst müssen Sie eine neue IAM-Richtlinie erstellen, damit CloudWatch Logs die `s3:PutObject` Berechtigung für den Amazon S3 S3-Ziel-Bucket erhält.

Um die Exportaufgabe in Schritt 5 erstellen zu können, müssen Sie mit der IAM-Rolle `AmazonS3ReadOnlyAccess` und bestimmten anderen Berechtigungen angemeldet sein. Sie können eine Richtlinie erstellen, die einige dieser anderen erforderlichen Berechtigungen enthält.

Welche Richtlinie 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.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-exported-logs/*"
        }
    ]
}
```

Wenn der Ziel-Bucket AWS KMS Verschlüsselung verwendet, erstellen Sie eine Richtlinie mit dem folgenden Inhalt.

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-exported-logs/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "ARN_OF_KMS_KEY"
        }
    ]
}
```

Um die Exportaufgabe in Schritt 5 zu erstellen, müssen Sie mit der IAM-Rolle `AmazonS3ReadOnlyAccess`, der IAM-Richtlinie, die Sie gerade erstellt haben, sowie mit den folgenden Berechtigungen angemeldet sein:

- `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 Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:

- Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer 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.

Schritt 3: Festlegen von Berechtigungen für einen S3-Bucket

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 Konto-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 `my-exported-logs` in den Namen Ihres S3-Bucket und `Principal` in den Endpunkt der Region, in der Sie Protokolldaten exportieren (beispielsweise `us-west-1`). Verwenden Sie einen Text-Editor, um diese Richtliniendatei zu erstellen. Verwenden Sie nicht die IAM-Konsole.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
    {
        "Action": "s3:GetBucketAcl",
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::my-exported-logs",
        "Principal": { "Service": "logs.Region.amazonaws.com" },
        "Condition": {
          "StringEquals": {
              "aws:SourceAccount": [
                  "AccountId1",
                  "AccountId2",
                  ...
              ]
          },
          "ArnLike": {
                  "aws:SourceArn": [
                      "arn:aws:logs:Region:AccountId1:log-group:*",
                      "arn:aws:logs:Region:AccountId2:log-group:*",
                      ...
                   ]
          }
        }
    },
    {
        "Action": "s3:PutObject" ,
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::my-exported-logs/*",
        "Principal": { "Service": "logs.Region.amazonaws.com" },
        "Condition": {
          "StringEquals": {
              "s3:x-amz-acl": "bucket-owner-full-control",
              "aws:SourceAccount": [
                  "AccountId1",
                  "AccountId2",
                  ...
              ]
          },
          "ArnLike": {
                  "aws:SourceArn": [
                      "arn:aws:logs:Region:AccountId1:log-group:*",
                      "arn:aws:logs:Region:AccountId2:log-group:*",
                      ...
                  ]
          }
        }
```

```
      },
      {
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::create_export_task_caller_account:role/role_name"
          },
          "Action": "s3:PutObject",
          "Resource": "arn:aws:s3:::my-exported-logs/*",
          "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control"
            }
          }
      }
    ]
}
```

2. Legen Sie die Richtlinie, die Sie gerade hinzugefügt haben, mithilfe des put-bucket-policyBefehls 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 my-exported-logs --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) Schritt 4: Export in einen mit SSE-KMS verschlüsselten Bucket

Dieser Schritt ist nur erforderlich, wenn Sie in einen 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. 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:

   - Ersetzen Sie *`Region`* durch die Region Ihrer Protokolle.
   - Ersetzen Sie *`Account-ARN`* (Konto-ARN) durch den ARN des Kontos, das Eigentümer des KMS-Schlüssels ist.

```
{
    "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": "*"
        },
```

```
        {
            "Sid": "Enable IAM Role Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS":
 "arn:aws:iam::create_export_task_caller_account:role/role_name"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "ARN_OF_KMS_KEY"
        }
    ]
}
```

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_id: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
    }
```

3. 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
    }
  ]
}
```

4. Geben Sie den folgenden Befehl ein und ersetzen Sie dabei *`bucket-name`* durch den Namen des Bucket, in den Sie Protokolle exportieren.

```
aws s3api put-bucket-encryption --bucket bucket-name --server-side-encryption-
configuration file://bucketencryption.json
```

   Wenn der Befehl keinen Fehler zurückgibt, ist der Vorgang erfolgreich abgeschlossen.

Schritt 5: Erstellen einer Exportaufgabe

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 Schritt 2: Einrichten von Zugriffsberechtigungen dokumentiert.
2. Verwenden Sie in einer Befehlszeile den folgenden create-export-taskBefehl, 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 "my-exported-logs" --destination-prefix "export-
task-output"
```

Es folgt eine Beispielausgabe.

```
{
    "taskId": "cda45419-90ea-4db5-9833-aade86253e66"
}
```

Beschreiben von Exportaufgaben

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-tasksBefehl.

```
aws logs --profile CWLExportUser 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 CWLExportUser describe-export-tasks --status-code "FAILED"
```

Es folgt eine Beispielausgabe.

```
{
   "exportTasks": [
   {
      "destination": "my-exported-logs",
      "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
   }]
}
```

Stornieren einer Exportaufgabe

Sie können eine Exportaufgabe stornieren, wenn sie entweder einen Status `PENDING` oder `RUNNING` aufweist.

Um eine Exportaufgabe abzubrechen, verwenden Sie AWS CLI

Verwenden Sie in einer Befehlszeile den folgenden cancel-export-taskBefehl:

```
aws logs --profile CWLExportUser cancel-export-task --task-id "cda45419-90ea-4db5-9833-aade86253e66"
```

Sie können den describe-export-tasksBefehl verwenden, um zu überprüfen, ob die Aufgabe erfolgreich abgebrochen wurde.

Streaming CloudWatch protokolliert Daten an Amazon OpenSearch Service

Sie können eine CloudWatch Logs-Protokollgruppe so konfigurieren, dass sie empfangene Daten über ein CloudWatch Logs-Abonnement nahezu in Echtzeit an Ihren Amazon OpenSearch Service-Cluster streamt. 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 unterKlassen protokollieren.

Abhängig von der Menge der gestreamten Protokolldaten möchten Sie möglicherweise eine Begrenzung der gleichzeitigen Ausführung der Funktion auf Funktionsebene festlegen. Weitere Informationen erhalten Sie unter Skalierung von Lambda-Funktionen.

> **Note**
>
> Das Streamen großer Mengen von CloudWatch Logs-Daten an den OpenSearch Service kann zu hohen Nutzungsgebühren führen. Wir empfehlen, dass Sie in der AWS Billing and Cost Management Konsole ein Budget erstellen. Weitere Informationen finden Sie unter Verwalten der Kosten mit AWS Budgets.

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:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "es:*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:es:region:account-id:domain/target-domain-name/
*"
        }
    ]
```

```
}
```

- Wenn die OpenSearch Zieldienstdomäne VPC-Zugriff verwendet, muss die AWSLambdaVPCAccessExecutionRoleRichtlinie 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 Servicefunktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien und serviceübergreifenden Beispiele sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie eine bestimmte Aufgabe ausführen können, indem Sie mehrere Funktionen innerhalb desselben Services aufrufen.

Serviceübergreifende Beispiele sind Beispielanwendungen, die über mehrere AWS-Services hinweg arbeiten.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter CloudWatch Logs mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Codebeispiele

- Aktionen für CloudWatch Protokolle, die AWS SDKs verwenden
  - Verwendung AssociateKmsKey mit einem AWS SDK oder CLI
  - Verwendung CancelExportTask mit einem AWS SDK oder CLI
  - Verwendung CreateExportTask mit einem AWS SDK oder CLI
  - Verwendung CreateLogGroup mit einem AWS SDK oder CLI
  - Verwendung CreateLogStream mit einem AWS SDK oder CLI
  - Verwendung DeleteLogGroup mit einem AWS SDK oder CLI
  - Verwendung DeleteSubscriptionFilter mit einem AWS SDK oder CLI
  - Verwendung DescribeExportTasks mit einem AWS SDK oder CLI
  - Verwendung DescribeLogGroups mit einem AWS SDK oder CLI
  - Verwendung DescribeSubscriptionFilters mit einem AWS SDK oder CLI
  - Verwendung GetQueryResults mit einem AWS SDK oder CLI
  - Verwendung PutSubscriptionFilter mit einem AWS SDK oder CLI
  - Verwendung StartLiveTail mit einem AWS SDK oder CLI
  - Verwendung StartQuery mit einem AWS SDK oder CLI

Aktionen für CloudWatch Protokolle, die AWS SDKs verwenden

Die folgenden Codebeispiele zeigen, wie einzelne CloudWatch Logs-Aktionen mit AWS SDKs ausgeführt werden. Diese Auszüge rufen die CloudWatch Logs-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon CloudWatch Logs API-Referenz.

Beispiele

- Verwendung AssociateKmsKey mit einem AWS SDK oder CLI
- Verwendung CancelExportTask mit einem AWS SDK oder CLI
- Verwendung CreateExportTask mit einem AWS SDK oder CLI
- Verwendung CreateLogGroup mit einem AWS SDK oder CLI
- Verwendung CreateLogStream mit einem AWS SDK oder CLI
- Verwendung DeleteLogGroup mit einem AWS SDK oder CLI
- Verwendung DeleteSubscriptionFilter mit einem AWS SDK oder CLI
- Verwendung DescribeExportTasks mit einem AWS SDK oder CLI
- Verwendung DescribeLogGroups mit einem AWS SDK oder CLI
- Verwendung DescribeSubscriptionFilters mit einem AWS SDK oder CLI
- Verwendung GetQueryResults mit einem AWS SDK oder CLI
- Verwendung PutSubscriptionFilter mit einem AWS SDK oder CLI
- Verwendung StartLiveTail mit einem AWS SDK oder CLI
- Verwendung StartQuery mit einem AWS SDK oder CLI

Verwendung **`AssociateKmsKey`** mit einem AWS SDK oder CLI

Das folgende Codebeispiel zeigt, wie es verwendet wird`AssociateKmsKey`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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/7c9eccc2-38cb-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 AssociateKmsKeyin der AWS SDK for .NET API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter CloudWatch Logs mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`CancelExportTask`** mit einem AWS SDK oder CLI

Das folgende Codebeispiel zeigt, wie es verwendet wird`CancelExportTask`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 CancelExportTaskin der AWS SDK for .NET API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter CloudWatch Logs mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`CreateExportTask`** mit einem AWS SDK oder CLI

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateExportTask`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 = "doc-example-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 CreateExportTaskin der AWS SDK for .NET API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter CloudWatch Logs mit einem AWS SDK verwenden. 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 Codebeispiele zeigen, wie es verwendet wird`CreateLogGroup`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 CreateLogGroupin der AWS SDK for .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 CreateLogGroupin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 CreateLogGroupin der AWS SDK for JavaScript API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter CloudWatch Logs mit einem AWS SDK verwenden. 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 Codebeispiele zeigen, wie es verwendet wird`CreateLogStream`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 CreateLogStreamin der AWS SDK for .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl erstellt einen Protokollstream mit dem Namen `20150601` in der Protokollgruppemy`-logs`:

```
aws logs create-log-stream --log-group-name my-logs --log-stream-name 20150601
```

- Einzelheiten zur API finden Sie CreateLogStreamin der AWS CLI Befehlsreferenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter CloudWatch Logs mit einem AWS SDK verwenden. 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 Codebeispiele zeigen, wie es verwendet wird`DeleteLogGroup`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 DeleteLogGroupin der AWS SDK for .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht eine Protokollgruppe mit dem Namen`my-logs`:

```
aws logs delete-log-group --log-group-name my-logs
```

- Einzelheiten zur API finden Sie DeleteLogGroupin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 DeleteLogGroupin der AWS SDK for JavaScript API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter CloudWatch Logs mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DeleteSubscriptionFilter`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`DeleteSubscriptionFilter`.

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 DeleteSubscriptionFilterin der AWS SDK for C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 DeleteSubscriptionFilterin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 DeleteSubscriptionFilterin der AWS SDK for JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 = {
  filterName: "FILTER",
  logGroupName: "LOG_GROUP",
};

cwl.deleteSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteSubscriptionFilterin der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun deleteSubFilter(filter: String?, logGroup: String?) {

    val request = DeleteSubscriptionFilterRequest {
        filterName = filter
        logGroupName = logGroup
    }

    CloudWatchLogsClient { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
 $filter")
    }
}
```

- Einzelheiten zur API finden Sie DeleteSubscriptionFilterin der API-Referenz zum AWS SDK für Kotlin.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter CloudWatch Logs mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DescribeExportTasks`** mit einem AWS SDK oder CLI

Das folgende Codebeispiel zeigt, wie es verwendet wird`DescribeExportTasks`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 DescribeExportTasksin der AWS SDK for .NET API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter CloudWatch Logs mit einem AWS SDK verwenden. 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 Codebeispiele zeigen, wie es verwendet wird`DescribeLogGroups`.

.NET

AWS SDK for .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 DescribeLogGroupsin der AWS SDK for .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 DescribeLogGroupsin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 && page.logGroups.every((lg) => !!lg)) {
      logGroups.push(...page.logGroups);
    }
  }

  console.log(logGroups);
  return logGroups;
};
```

- Einzelheiten zur API finden Sie DescribeLogGroupsin der AWS SDK for JavaScript API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter CloudWatch Logs mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DescribeSubscriptionFilters`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`DescribeSubscriptionFilters`.

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 DescribeSubscriptionFiltersin der AWS SDK for C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 DescribeSubscriptionFiltersin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 DescribeSubscriptionFiltersin der AWS SDK for JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DescribeSubscriptionFiltersin der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

```
suspend fun describeFilters(logGroup: String) {

    val request = DescribeSubscriptionFiltersRequest {
        logGroupName = logGroup
        limit = 1
    }

    CloudWatchLogsClient { 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 DescribeSubscriptionFiltersin der API-Referenz zum AWS SDK für Kotlin.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter CloudWatch Logs mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`GetQueryResults`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`GetQueryResults`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Führen Sie eine umfangreiche Abfrage aus

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 GetQueryResultsin der AWS SDK for JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 GetQueryResultsin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. CloudWatch Logs mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`PutSubscriptionFilter`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`PutSubscriptionFilter`.

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 PutSubscriptionFilterin der AWS SDK for C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 PutSubscriptionFilterin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 PutSubscriptionFilterin der AWS SDK for JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 = {
  destinationArn: "LAMBDA_FUNCTION_ARN",
  filterName: "FILTER_NAME",
  filterPattern: "ERROR",
  logGroupName: "LOG_GROUP",
};

cwl.putSubscriptionFilter(params, function (err, data) {
```

```
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK for JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutSubscriptionFilterin der AWS SDK for JavaScript API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter CloudWatch Logs mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`StartLiveTail`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`StartLiveTail`.

.NET

AWS SDK for .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 der Live-Tail-Sitzung auf zwei Arten behandeln:

```
            /* 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 StartLiveTailin der AWS SDK for .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"
)
```

Behandeln 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 occured 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 StartLiveTailin der AWS SDK for 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;
```

Behandeln 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 StartLiveTailin der 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";
```

Behandelt 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 StartLiveTailin der AWS SDK for 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 unsubcribe 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 StartLiveTailin 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 StartLiveTailin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. CloudWatch Logs mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`StartQuery`** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie es verwendet wird`StartQuery`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Führen Sie eine umfangreiche Abfrage aus

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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 StartQueryin der AWS SDK for JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository 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_groups,
                    startTime=start_time,
                    endTime=end_time,
                    queryString="fields @timestamp, @message | sort @timestamp
asc",
                    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_groups,
                startTime=start_time,
                endTime=end_time,
                queryString="fields @timestamp, @message | sort @timestamp asc",
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")
```

- Einzelheiten zur API finden Sie StartQueryin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. CloudWatch Logs mit einem AWS SDK verwenden Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Szenarien für CloudWatch Protokolle, die AWS SDKs verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in CloudWatch Logs with AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen in CloudWatch Logs aufrufen. Jedes Szenario enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Beispiele

- Verwenden Sie CloudWatch Logs, um eine umfangreiche Abfrage auszuführen

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.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Das 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(parseInt(process.env.QUERY_START_DATE)),
    new Date(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]);
  }

  // snippet-start:[javascript.v3.cloudwatch-logs.actions.GetQueryResults]
  /**
   * Simple wrapper for the GetQueryResultsCommand.
   * @param {string} queryId
   */
  _getQueryResults(queryId) {
    return this.client.send(new GetQueryResultsCommand({ queryId }));
  }
  // snippet-end:[javascript.v3.cloudwatch-logs.actions.GetQueryResults]

  /**
   * 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 [];
      } else {
        throw err;
      }
    }
  }

  // snippet-start:[javascript.v3.cloudwatch-logs.actions.StartQuery]
  /**
   * 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;
    }
  }
  // snippet-end:[javascript.v3.cloudwatch-logs.actions.StartQuery]

  /**
   * 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;
      },
    );
  }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for JavaScript -API-Referenz.
  - GetQueryResults
  - StartQuery

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel-Repository einrichten und ausführen.

Diese Datei ruft ein Beispielmodul für die Verwaltung von CloudWatch Abfragen mit mehr als 10.000 Ergebnissen auf.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
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",
)


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.

        :return: Tuple of query start date and end date as integers.
        :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)
```

```
        return query_start_date, query_end_date

    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",
    ):
        """
        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
        """
        cloudwatch_query = CloudWatchQuery(
            [start_date_iso8601, end_date_iso8601],
        )
```

```
        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 = 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)


if __name__ == "__main__":
    main()
```

Dieses Modul verarbeitet CloudWatch Abfragen mit mehr als 10.000 Ergebnissen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import logging
import time
from datetime import datetime
import threading
import boto3

from date_utilities import DateUtilities


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.

    :ivar date_range: Start and end datetime for the query.
    :vartype date_range: tuple
    :ivar limit: Maximum number of log entries to return.
    :vartype limit: int
    """

    def __init__(self, date_range):
        self.lock = threading.Lock()
        self.log_groups = "/workflows/cloudwatch-logs/large-query"
        self.query_results = []
        self.date_range = date_range
        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}"
        )
```

```
        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

    # snippet-start:[python.example_code.cloudwatch_logs.start_query]
    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_groups,
                    startTime=start_time,
                    endTime=end_time,
                    queryString="fields @timestamp, @message | sort @timestamp
asc",
                    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_groups,
                startTime=start_time,
                endTime=end_time,
                queryString="fields @timestamp, @message | sort @timestamp asc",
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")

    # snippet-end:[python.example_code.cloudwatch_logs.start_query]

    # snippet-start:[python.example_code.cloudwatch_logs.get_query_results]
    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", [])

    # snippet-end:[python.example_code.cloudwatch_logs.get_query_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 CloudWatch Logs mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Serviceübergreifende Beispiele für CloudWatch Logs, die SDKs verwenden AWS

Die folgenden Beispielanwendungen verwenden AWS SDKs, um CloudWatch Logs mit anderen zu kombinieren. AWS-Services Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen der Anwendung finden.

Beispiele

- Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

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.

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
- EventBridge
- Lambda

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter CloudWatch Logs mit einem AWS SDK verwenden. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

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

> **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 Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie 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).
- Verwenden Sie SSL/TLS, um mit Ressourcen zu kommunizieren. AWS Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein. AWS CloudTrail
- 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 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

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 serverseitigen Verschlüsselungsschlüssel.

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 die end-to-end 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 Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer 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 Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer 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

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
Protokollgruppe	Die folgenden beiden werden verwendet. Die zweite, mit dem `:*` am Ende, wird vom `describe-log-groups` CLI-Befehl und der DescribeLogGroupsAPI zurückgegeben. arn:aws:logs:*region*:*account-id* :log-group:*log_group_name* arn:aws:logs:*region*:*account-id* :log-group:*log_group_name* :* Verwenden Sie die erste Version ohne das Ende `:*` in den folgenden Situationen: • In vielen CloudWatch Logs APIs im `logGroupIdentifier` Eingabefeld. • Im `resourceArn` Feld beim Taggen von APIs • In IAM Richtlinien, wenn Berechtigungen für TagResourceUntagResource, und ListTagsForResourceangegeben werden. Verwenden Sie die zweite Version mit dem Ende`:*`, um bei der Angabe von Berechtigungen in IAM-Richtlinien für alle anderen API-Aktionen auf den ARN zu verweisen.
Protokoll-Stream	*arn:aws:logs: region: account-id:log-group: log_group_name:log-stream: log-stream-name*
Bestimmungsort	arn:aws:logs:*region*:*account-id* :destination:*destination_name*

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 Prinzipalentität (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.

Verwalten 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.

Mit einer IAM-Identität verknüpfte Richtlinien 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
- Ressourcenbasierte 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 zum Analysieren von Daten mit hoher Kardinalität.

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 dann die Ergebnisse anzeigt. Die Ergebnisse können Contributor-Daten für diese Protokollgruppen enthalten. Gewähren Sie diese Berechtigungen nur Benutzern, die diese Daten anzeigen können sollten.

Ressourcenbasierte 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 PutDestinationAPI erstellt werden, und Sie können dem Ziel mithilfe der PutDestinationAPI eine Ressourcenrichtlinie hinzufügen. Im folgenden Beispiel kann ein anderes AWS Konto mit der Konto-ID 111122223333 seine Protokollgruppen für das Ziel abonnieren. `arn:aws:logs:us-east-1:123456789012:destination:testDestination`

```
{
  "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 Richtliniensyntax 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 Richtlinienschlüssel 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, mit denen Sie den Zugriff auf Ihre CloudWatch Logs-Ressourcen verwalten können. 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:

```
{
  "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 (*) am Ende des `Resource`-Werts bedeutet, dass die Anweisung Berechtigungen für die `logs:CreateLogGroup`-, `logs:CreateLogStream`-, `logs:PutLogEvents`- und `logs:DescribeLogStreams`-Aktionen einer Protokollgruppe zulässt. Um diese Berechtigungen auf eine bestimmte Protokollgruppe zu beschränken, ersetzen Sie das Platzhalterzeichen (*) im ARN der Ressource durch den spezifischen ARN der Protokollgruppe. Weitere Informationen über die Abschnitte in einer IAM-Richtlinienanweisung finden Sie in der Referenz zu IAM-Richtlinienelemente im IAM-Benutzerhandbuch. Eine Liste mit allen CloudWatch Logs-Aktionen finden Sie unterCloudWatch Referenz zu Protokollberechtigungen.

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 beschriebenAWS 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 beinhaltet die `cloudwatch:GenerateQuery` Berechtigung, sodass Benutzer mit dieser Richtlinie anhand einer Aufforderung in natürlicher Sprache eine CloudWatch Logs Insights-Abfragezeichenfolge generieren können. Der Inhalt ist wie folgt:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:*",
                "cloudwatch:GenerateQuery"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

CloudWatchLogsReadOnlyAccess

Die CloudWatchLogsReadOnlyAccessRichtlinie gewährt nur Lesezugriff auf Logs. CloudWatch Sie beinhaltet die `cloudwatch:GenerateQuery` Berechtigung, sodass Benutzer mit dieser

Richtlinie anhand einer Aufforderung in natürlicher Sprache eine CloudWatch Logs Insights-Abfragezeichenfolge generieren können. Der Inhalt ist wie folgt:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:Describe*",
                "logs:Get*",
                "logs:List*",
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:TestMetricFilter",
                "logs:FilterLogEvents",
                "logs:StartLiveTail",
                "logs:StopLiveTail",
                "cloudwatch:GenerateQuery"
            ],
            "Resource": "*"
        }
    ]
}
```

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 CloudWatch kontoübergreifende Observability.

Der Inhalt ist wie folgt:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:Link",
                "oam:ListLinks"
            ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:DeleteLink",
                "oam:GetLink",
                "oam:TagResource"
            ],
            "Resource": "arn:aws:oam:*:*:link/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink"
            ],
            "Resource": [
                "arn:aws:oam:*:*:link/*",
                "arn:aws:oam:*:*:sink/*"
            ]
        }
    ]
}
```

CloudWatch Protokolliert Aktualisierungen verwalteter Richtlinien AWS

Details zu Aktualisierungen der AWS verwalteten Richtlinien für CloudWatch Logs anzeigen, 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.

Änderung	Beschreibung	Datum
CloudWatchLogsFullAccess – Aktualisierung auf eine bestehende Richtlinie.	CloudWatch Logs hat eine Berechtigung zu hinzugefügt CloudWatchLogsFullAccess. Die `cloudwatch:GenerateQuery` Berechtigung wurde	8. November 2023

Änderung	Beschreibung	Datum
	hinzugefügt, sodass Benutzer mit dieser Richtlinie anhand einer Aufforderung in natürlicher Sprache eine CloudWatch Logs Insights-Abfragesprache generieren können.	
CloudWatchLogsReadOnlyAccess – Aktualisierung auf eine bestehende Richtlinie.	CloudWatch hat eine Berechtigung zu hinzugefügt CloudWatchLogsReadOnlyAccess. Die `cloudwatch:GenerateQuery` Berechtigung wurde hinzugefügt, sodass Benutzer mit dieser Richtlinie anhand einer Aufforderung in natürlicher Sprache eine CloudWatch Logs Insights-Abfragesprache generieren können.	8. November 2023

Änderung	Beschreibung	Datum
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 Observability CloudWatch	27. November 2022

Änderung	Beschreibung	Datum
CloudWatchLogsRead OnlyAccess – Aktualisierung auf eine bestehende Richtlinie	CloudWatch Protokolliert hinzugefügte Berechtigungen für. CloudWatchLogsRead OnlyAccess Die `oam:ListAttachedLinks` Berechtigungen `oam:ListSinks` 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: Zugriff auf eine einzelne Protokollgruppe erlauben

Beispiel 1: Vollzugriff auf Logs zulassen CloudWatch

Die folgende Richtlinie ermöglicht einem Benutzer den Zugriff auf alle CloudWatch Logs-Aktionen.

```
{
  "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.

```
{
    "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: Zugriff auf eine einzelne Protokollgruppe erlauben

Mit der folgenden Richtlinie kann ein Benutzer Protokollereignisse in einer bestimmten Protokollgruppe lesen und schreiben.

> **Important**
>
> Das `:*` am Ende des Protokollgruppennamens in der `Resource`-Zeile ist erforderlich, um anzuzeigen, dass die Richtlinie für alle Protokollabläufe in dieser Protokollgruppe gilt. Wenn Sie `:*` weglassen, wird die Richtlinie nicht durchgesetzt.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
      "Action": [
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "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`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:*"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/Team": "Green"
                }
            }
        }
    ]
}
```

Die Operationen StopQueryund 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.

```
{
    "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	`logs:CancelExportTask` Erforderlich zum Abbrechen einer ausstehen den oder laufenden Exportaufgabe.
CreateExportTask	`logs:CreateExportTask`

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
	Erforderlich zum Exportieren von Daten aus einer Protokollgruppe in einen Amazon-S3-Bucket.
CreateLogGroup	`logs:CreateLogGroup` Erforderlich zum Erstellen einer neuen Protokoll gruppe.
CreateLogStream	`logs:CreateLogStream` Erforderlich zum Erstellen eines neuen Protokoll-Stream in eine Protokollgruppe.
DeleteDestination	`logs:DeleteDestination` Erforderlich zum Löschen eines Protokollziels und Aktivieren von Abonnementfiltern für das Ziel.
DeleteLogGroup	`logs:DeleteLogGroup` Erforderlich zum Löschen einer Protokoll gruppe und der jeweiligen archivierten Protokoll ereignisse.
DeleteLogStream	`logs:DeleteLogStream` Erforderlich zum Löschen eines Protokoll-Stream und der jeweiligen archivierten Protokoll ereignisse.
DeleteMetricFilter	`logs:DeleteMetricFilter` Erforderlich zum Löschen eines Metrikfilters für eine Protokollgruppe.

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
DeleteQueryDefinition	`logs:DeleteQueryDefinition` Erforderlich, um eine gespeicherte Abfragede finition in CloudWatch Logs Insights zu löschen.
DeleteResourcePolicy	`logs:DeleteResourcePolicy` Erforderlich, um eine CloudWatch Logs-Ress ourcenrichtlinie zu löschen.
DeleteRetentionPolicy	`logs:DeleteRetentionPolicy` Erforderlich zum Löschen der Aufbewahr ungsrichtlinie einer Protokollgruppe.
DeleteSubscriptionFilter	`logs:DeleteSubscriptionFilter` Erforderlich zum Löschen des Abonnemen tfilters für eine Protokollgruppe.
DescribeDestinations	`logs:DescribeDestinations` Erforderlich zum Anzeigen aller Ziele für ein Konto.
DescribeExportTasks	`logs:DescribeExportTasks` Erforderlich zum Anzeigen aller Exportauf gaben für das Konto.
DescribeLogGroups	`logs:DescribeLogGroups` Erforderlich zum Anzeigen aller Protokoll gruppen für ein Konto.
DescribeLogStreams	`logs:DescribeLogStreams` Erforderlich zum Anzeigen aller Protokoll-Streams für eine Protokollgruppe.

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
DescribeMetricFilters	`logs:DescribeMetricFilters` Erforderlich zum Anzeigen aller Metriken für eine Protokollgruppe.
DescribeQueryDefinitions	`logs:DescribeQueryDefinitions` Erforderlich, um die Liste der gespeicherten Abfragedefinitionen in CloudWatch Logs Insights zu sehen.
DescribeQueries	`logs:DescribeQueries` Erforderlich, um die Liste der CloudWatch Logs Insights-Abfragen zu sehen, die geplant sind, ausgeführt werden oder kürzlich ausgeführt wurden.
DescribeResourcePolicies	`logs:DescribeResourcePolicies` Erforderlich, um eine Liste der CloudWatch Logs-Ressourcenrichtlinien anzuzeigen.
DescribeSubscriptionFilters	`logs:DescribeSubscriptionFilters` Erforderlich zum Anzeigen aller Abonnemen tfilter für eine Protokollgruppe.
FilterLogEvents	`logs:FilterLogEvents` Erforderlich zum Sortieren von Protokoll ereignissen nach Gruppenfiltermuster.
GetLogEvents	`logs:GetLogEvents` Erforderlich zum Abrufen von Protokollereigniss en aus einem Protokoll-Stream.

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
GetLogGroupFields	`logs:GetLogGroupFields` Erforderlich, um die Liste der Felder abzurufen , die in den Protokollereignissen in einer Protokollgruppe enthalten sind.
GetLogRecord	`logs:GetLogRecord` Erforderlich, um die Details aus einem einzelnen Protokollereignis abzurufen.
GetQueryResults	`logs:GetQueryResults` Erforderlich, um die Ergebnisse von CloudWatch Logs Insights-Abfragen abzurufen.
ListTagsLogGroup	`logs:ListTagsLogGroup` Erforderlich zum Auflisten der mit einer Protokollgruppe verbundenen Tags.
PutDestination	`logs:PutDestination` Erforderlich zum Erstellen oder Aktualisieren eines Ziel-Protokoll-Streams (z. B. ein Kinesis-Stream).
PutDestinationPolicy	`logs:PutDestinationPolicy` Erforderlich zum Erstellen oder Aktualisieren einer Zugriffsrichtlinie im Zusammenhang mit einem vorhandenen Protokollziel.
PutLogEvents	`logs:PutLogEvents` Erforderlich für den Upload eines Batches von Protokollereignissen in einen Protokoll-Stream.

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
PutMetricFilter	`logs:PutMetricFilter` Erforderlich zum Erstellen oder Aktualisieren eines Metrikfilters, der einer Protokollgruppe zugeordnet wird.
PutQueryDefinition	`logs:PutQueryDefinition` Erforderlich, um eine Abfrage in CloudWatch Logs Insights zu speichern.
PutResourcePolicy	`logs:PutResourcePolicy` Erforderlich, um eine CloudWatch Logs-Ressourcenrichtlinie zu erstellen.
PutRetentionPolicy	`logs:PutRetentionPolicy` Erforderlich zum Festlegen der Anzahl der Tage, die Protokollereignisse (Aufbewahrung) in einer Protokollgruppe aufbewahrt werden sollen.
PutSubscriptionFilter	`logs:PutSubscriptionFilter` Erforderlich zum Erstellen oder Aktualisieren eines Abonnementfilters, der einer Protokoll gruppe zugeordnet wird.
StartQuery	`logs:StartQuery` Erforderlich, um CloudWatch Logs Insights-Abfragen zu starten.
StopQuery	`logs:StopQuery` Erforderlich, um eine laufende CloudWatch Logs Insights-Abfrage zu beenden.

CloudWatch Protokolliert API-Operationen	Erforderliche Berechtigungen (API-Aktionen)
TagLogGroup	`logs:TagLogGroup` Erforderlich zum Hinzufügen oder Aktualisieren von Protokollgruppen-Tags.
TestMetricFilter	`logs:TestMetricFilter` 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 Dienstbezogene Rollen sind in CloudWatch Logs vordefiniert und beinhalten 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 nach den Services, für die Yes (Ja) in der Spalte Serviceverknüpfte Rolle angegeben ist. Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Dienstbezogene Rollenberechtigungen für Logs CloudWatch

CloudWatch Logs verwendet die angegebene dienstverknüpfte Rolle. AWSServiceRoleForLogDelivery CloudWatch Logs verwendet diese dienstbezogene Rolle, um Protokolle direkt in Firehose zu schreiben. Weitere Informationen finden Sie unter Aktivieren der Protokollierung von AWS Diensten.

Die serviceverknüpfte Rolle AWSServiceRoleForLogDelivery vertraut darauf, dass die folgenden Services die Rolle annehmen:

- `logs.amazonaws.com`

Die Richtlinie für Rollenberechtigungen ermöglicht es CloudWatch Logs, die folgenden Aktionen an den angegebenen Ressourcen durchzuführen:

- Aktion: `firehose:PutRecord` und `firehose:PutRecordBatch` auf allen Firehose-Streams, die ein Tag mit einem `LogDeliveryEnabled` Schlüssel mit einem Wert von `True` haben. Dieses Tag wird automatisch an einen Firehose-Stream angehängt, wenn Sie ein Abonnement für die Übermittlung der Protokolle an Firehose erstellen.

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 serviceverknüpfte 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 Management Console, 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 Rollenname nach der Erstellung nicht geändert werden. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen einer dienstbezogenen 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 von der mit dem AWSServiceRoleForLogDeliveryDienst verknüpften Rolle verwendet werden

- 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 AWSServiceRoleForLogDeliveryserviceverknüpfte Rolle zu löschen. Weitere Informationen finden Sie unter Löschen einer serviceverknüpften Rolle

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.

Compliance-Validierung für Amazon CloudWatch Logs

Externe Prüfer bewerten im Rahmen verschiedener AWS -Compliance-Programme die Sicherheit und Compliance von Amazon CloudWatch Logs. Hierzu zählen unter anderem SOC, PCI, FedRAMP und HIPAA.

Eine Liste der - AWS Services, die in den Geltungsbereich bestimmter Compliance-Programme fallen, finden Sie unter AWS -Services im Geltungsbereich nach Compliance-Programm Allgemeine Informationen finden Sie unter AWS -Compliance-Programme.

Sie können Auditberichte von Drittanbietern mit herunterladen AWS Artifact. Weitere Informationen finden Sie unter Herunterladen von Berichten unter AWS Artifact .

Ihre Compliance-Verantwortung bei der Verwendung von Amazon CloudWatch Logs hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Unterstützung der Compliance bereit:

- Schnellstartanleitungen für Sicherheit und Compliance – In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Bereitstellung von sicherheits- und konformitätsorientierten Basisumgebungen auf AWSangegeben.
- Architekturerstellung für HIPAA-Sicherheit und -Compliance in Amazon Web Services – In diesem Whitepaper wird beschrieben, wie Unternehmen mithilfe AWS von HIPAA-konforme Anwendungen erstellen können.
- AWS Compliance-Ressourcen – Diese Sammlung von Arbeitsmappen und Leitfäden könnte für Ihre Branche und Ihren Standort gelten.
- Bewertung von Ressourcen mit Regeln im -AWS Config Entwicklerhandbuch AWS Config– bewertet, wie gut Ihre Ressourcenkonfigurationen internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub – Dieser AWS Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus in AWS , mit dem Sie Ihre Compliance mit den Sicherheitsstandards und bewährten Methoden der Branche überprüfen können.

Ausfallsicherheit in Amazon CloudWatch Logs

Im Zentrum der globalen AWS-Infrastruktur stehen die 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 hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen über AWS-Regionen und -Availability Zones finden Sie unter Globale AWS-Infrastruktur.

Infrastruktursicherheit in Amazon CloudWatch Logs

Als verwalteter Service ist Amazon CloudWatch Logs durch die AWS globale Netzwerksicherheit von geschützt. Informationen zu AWS Sicherheitsservices und wie die Infrastruktur AWS schützt, finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung mit den bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastrukturschutz in Security Pillar AWS Well-Architected Framework.

Sie verwenden durch 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.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Verwenden von CloudWatch Protokollen mit Schnittstellen-VPC-Endpunkten

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 CloudWatch Logs zu senden, ohne sie über das Internet zu senden.

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 Protokollen zu verbinden, definieren Sie einen Schnittstellen-VPC-Endpunkt für CloudWatch Protokolle. Mit dieser Art Endpunkt können Sie Ihre VPC mit AWS -

Services verbinden. Der Endpunkt bietet zuverlässige, skalierbare Konnektivität zu - CloudWatch Protokollen, ohne dass ein Internet-Gateway, eine NAT-Instance (Network Address Translation) oder eine VPN-Verbindung erforderlich ist. Weitere Informationen finden Sie unter Was ist Amazon VPC im Benutzerhandbuch zu Amazon VPC.

Schnittstellen-VPC-Endpunkte werden von unterstützt AWS PrivateLink, einer - AWS Technologie, die private Kommunikation zwischen - AWS Services über eine Elastic-Network-Schnittstelle mit privaten IP-Adressen ermöglicht. Weitere Informationen finden Sie unter Neu – AWS PrivateLink für - AWS Services.

Die folgenden Schritte sind für Benutzer von Amazon VPC vorgesehen. Weitere Informationen finden Sie unter Erste Schritte im Amazon VPC Benutzerhandbuch.

Verfügbarkeit

CloudWatch Protokolle unterstützen derzeit VPC-Endpunkte in allen - AWS Regionen, einschließlich der - AWS GovCloud (US) Regionen.

Erstellen eines VPC-Endpunkts für CloudWatch Protokolle

Um mit der Verwendung von CloudWatch Protokollen mit Ihrer VPC zu beginnen, erstellen Sie einen Schnittstellen-VPC-Endpunkt für CloudWatch Protokolle. Der auszuwählende Service ist com.amazonaws.***Region***.logs. Sie müssen keine Einstellungen für CloudWatch Protokolle ändern. Weitere Informationen finden Sie unter Erstellen eines Schnittstellenendpunkts im Amazon VPC Leitfaden.

Testen der Verbindung zwischen Ihrer VPC und - CloudWatch Protokollen

Nachdem Sie den Endpunkt erstellt haben, können Sie die Verbindung testen.

So testen Sie die Verbindung zwischen Ihrer VPC und Ihrem CloudWatch Logs-Endpunkt

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 die , 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 Protokolle. Diese Richtlinie ermöglicht es Benutzern, eine Verbindung zu CloudWatch Protokollen über die VPC herzustellen, Protokollstreams zu erstellen und Protokolle an CloudWatch Protokolle zu senden, und verhindert, dass sie andere CloudWatch Protokollaktionen ausführen.

```
{
  "Statement": [
    {
      "Sid": "PutOnly",
      "Principal": "*",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

So ändern Sie die VPC-Endpunktrichtlinie für CloudWatch Protokolle

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 noch nicht für CloudWatch Protokolle erstellt haben, wählen Sie Endpunkt erstellen aus. Wählen Sie anschließend com.amazonaws.***Region***.logs und danach Create endpoint (Endpunkt erstellen) aus.
4. Wählen Sie den Endpunkt com.amazonaws.***Region***.logs und danach die Registerkarte Policy (Richtlinie) in der unteren Hälfte des Bildschirms aus.
5. Wählen Sie Richtlinie bearbeiten und nehmen Sie die Änderungen an der Richtlinie vor.

Support für VPC-Kontextschlüssel

CloudWatch Logs unterstützt die `aws:SourceVpce` Kontextschlüssel `aws:SourceVpc` und , die den Zugriff auf bestimmte VPCs oder bestimmte 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.

Protokollieren von Amazon-CloudWatch-Logs-API-Aufrufen in AWS CloudTrail

Amazon CloudWatch Logs ist in AWS CloudTrail integriert. Dieser Service zeichnet die Aktionen eines Benutzers, einer Rolle oder eines AWS-Service in CloudWatch Logs auf. CloudTrail erfasst API-Aufrufe, die vom oder für Ihr AWS-Konto getätigt wurden. Zu den erfassten Aufrufen gehören Aufrufe von der CloudWatch-Konsole und Code-Aufrufe an die CloudWatch-Logs-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail-Ereignissen an einen Amazon-S3-Bucket aktivieren, einschließlich Ereignissen für CloudWatch Logs. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse in der CloudTrail-Konsole trotzdem in Event history (Ereignisverlauf) anzeigen. Mit den von CloudTrail erfassten Informationen können Sie die an CloudWatch Logs gestellte Anfrage, die IP-Adresse, von der die Anfrage gestellt wurde, den Initiator der Anfrage, den Zeitpunkt der Anfrage und zusätzliche Details bestimmen.

Weitere Informationen über CloudTrail, einschließlich Konfiguration und Aktivierung, finden Sie im AWS CloudTrail-Benutzerhandbuch.

Themen

CloudWatch-Logs-Informationen in CloudTrail

CloudTrail wird beim Erstellen Ihres AWS-Kontos für Sie aktiviert. Wenn die unterstützte Ereignisaktivität in CloudWatch Logs eintritt, wird diese Aktivität in einem CloudTrail-Ereignis zusammen mit anderen AWS-Serviceereignissen im Ereignisverlauf aufgezeichnet. Sie können die neusten Ereignisse in Ihr AWS-Konto herunterladen und dort suchen und anzeigen. Weitere Informationen finden Sie unter Anzeigen von Ereignissen mit dem CloudTrail-Ereignisverlauf.

Erstellen Sie einen Trail für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS-Konto, inklusive Ereignisse für CloudWatch Logs. Ein Trail ermöglicht es CloudTrail, Protokolldateien in einem Amazon-S3-Bucket bereitzustellen. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen in der AWS-Partition und stellt die Protokolldateien in dem von Ihnen angegebenen Amazon S3 Bucket bereit. Darüber hinaus können Sie andere AWS-Services konfigurieren, um die in den CloudTrail-Protokollen erfassten

Ereignisdaten weiter zu analysieren und entsprechend zu agieren. Weitere Informationen finden Sie unter:

- Übersicht zum Erstellen eines Trails
- Von CloudTrail unterstützte Dienste und Integrationen
- Konfigurieren von Amazon-SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail-Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail-Protokolldateien aus mehreren Konten

CloudWatch Logs unterstützt die Protokollierung der folgenden Aktionen als Ereignisse in CloudTrail-Protokolldateien:

- CancelExportTask
- CreateExportTask
- CreateLogGroup
- CreateLogStream
- DeleteDestination
- DeleteLogGroup
- DeleteLogStream
- DeleteMetricFilter
- DeleteRetentionPolicy
- DeleteSubscriptionFilter
- PutDestination
- PutDestinationPolicy
- PutMetricFilter
- PutResourcePolicy
- PutRetentionPolicy
- PutSubscriptionFilter
- StartQuery
- StopQuery
- TestMetricFilter

Nur Anforderungselemente werden für diese CloudWatch-Logs-API-Aktionen in CloudTrail protokolliert:

- DescribeDestinations
- DescribeExportTasks
- DescribeLogGroups
- DescribeLogStreams
- DescribeMetricFilters
- DescribeQueries
- DescribeResourcePolicies
- DescribeSubscriptionFilters
- FilterLogEvents
- GetLogEvents
- GetLogGroupFields
- GetLogRecord
- GetQueryResults

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 Anfrage mit Root- oder IAM-Benutzer-Anmeldeinformationen von ausgeführt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen verbundenen Benutzer gesendet wurde.
- Gibt an, ob die Anforderung aus einem anderen AWS-Service gesendet wurde

Weitere Informationen finden Sie unter dem CloudTrail userIdentity-Element.

Grundlagen zu -Protokolldateieinträgen

Ein Trail ist eine Konfiguration, durch die Ereignisse als Protokolldateien an den von Ihnen angegebenen Amazon-S3-Bucket übermittelt werden. CloudTrail-Protokolldateien können einen oder mehrere Einträge enthalten. Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält unter anderem Informationen über die angeforderte Aktion, das Datum und die Uhrzeit

der Aktion sowie über die Anfrageparameter. CloudTrail-Protokolleinträge sind kein geordnetes Stack-Trace der öffentlichen API-Aufrufe und erscheinen daher in keiner bestimmten Reihenfolge.

Der folgende Protokolldateieintrag zeigt, dass ein Benutzer die CloudWatch-Logs-Aktion CreateExportTask 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"
}
```

Referenz zum CloudWatch-Logs-Agenten

> **Important**
>
> Diese Referenz gilt für den älteren, überholten CloudWatch-Logs-Agenten. Wenn Sie Instance-Metadaten-Service-Version 2 (IMDSv2) verwenden, müssen Sie den neuen einheitlichen CloudWatch-Agenten verwenden. Auch wenn Sie IMDSv2 nicht verwenden, empfehlen wir dringend, den neueren einheitlichen CloudWatch-Agenten anstelle des älteren Protokollagenten zu verwenden. Weitere Informationen über diesen neueren einheitlichen Agenten finden Sie unter Erfassen von Metriken und Protokollen von Amazon-EC2-Instances und On-Premises-Servern mit dem CloudWatch-Agent.
> Weitere Informationen zur Migration vom älteren CloudWatch-Logs-Agenten zum einheitlichen Agenten finden Sie unter Erstellen der Konfigurationsdatei des CloudWatch-Agenten mit dem Assistenten.

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

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

Agent-Konfigurationsdatei

Die CloudWatch-Logs-Agent-Konfigurationsdatei beschreibt Informationen, die für den CloudWatch-Logs-Agent erforderlich sind. 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 ist im Python-Format für Konfigurationsdateien (https://docs.python.org/2/library/logging.config.html#logging-config-fileformat) geschrieben. 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 dieser Wert auf true (Standard) gesetzt ist, kann HTTP-Inhaltsverschlüsselung GZIP-komprimierte Nutzlasten an CloudWatch Logs senden. Dies verringert die CPU-Auslastung, reduziert NetworkOut und senkt die Put-Latenz. Um diese Funktion zu deaktivieren, fügen Sie use_gzip_http_content_encoding = false im Abschnitt [general] der CloudWatch-Logs-Agent-Konfigurationsdatei ein. Starten Sie dann den Agent neu.

> **Note**
>
> Diese Einstellung ist nur awscli-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 "." (Punkt).

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 Zeitzonenversatz (%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 die Protokolldateien an, die an CloudWatch Logs übertragen werden sollen. Die Datei kann auf eine bestimmte Datei oder mehrere Dateien (mit Platzhaltern wie /var/log/system.log*)

verweisen. Nur die aktuelle Datei wird basierend auf dem Änderungsdatum der Datei an 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. Fingerabdruck-Zeilen werden nicht an CloudWatch Logs gesendet, es sei denn, alle angegebenen Zeilen sind verfügbar.

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.

Verwendung des CloudWatch-Logs-Agenten mit HTTP-Proxys

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

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

Verwendung des CloudWatch-Logs-Agent mit HTTP-Proxys

1. Führen Sie eine der folgenden Aktionen aus:

   a. Für eine Neuinstallation des CloudWatch-Logs-Agent, führen Sie 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-Metadatenservice 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-Benutzerhandbuch für Linux-Instances.

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

   b. Zum Bearbeiten einer vorhandenen Installation des CloudWatch-Logs-Agent fügen Sie die Proxys in /var/awslogs/etc/proxy.conf 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
   ```

Aufgliedern der Konfigurationsdateien für den CloudWatch-Logs-Agenten

Wenn Sie awslogs-agent-setup.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 /var/awslogs/etc/config/ erstellen. Wenn der CloudWatch-Logs-Agent gestartet wird, umfasst er sämtliche Stream-Konfigurationen in diesen zusätzlichen Konfigurationsdateien. Konfigurationseigenschaften im Abschnitt [general] müssen in der Haupt-Konfigurationsdatei (/var/awslogs/etc/awslogs.conf) definiert werden und werden in den anderen unter /var/awslogs/etc/config/ vorhandenen Konfigurationsdateien ignoriert.

Wenn Sie kein /var/awslogs/etc/config/-Verzeichnis 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
```

Häufig gestellte Fragen zum CloudWatch-Logs-Agenten

Welche Arten von Dateirotationen werden unterstützt?

Folgende Dateirotationsmechanismen werden unterstützt:

- Umbenennen vorhandener Protokolldateien mit einem numerischen Suffix, anschließendes Neuerstellen der ursprünglichen leeren Protokolldatei. Z. B. /var/log/syslog.log is renamed /var/log/syslog.log.1. Wenn /var/log/syslog.log.1 bereits aus einem vorherigen Durchgang vorhanden ist, wird es in /var/log/syslog.log.2 umbenannt.
- Kürzen der ursprünglichen Protokolldatei nach der Erstellung einer Kopie. Beispielsweise wird /var/log/syslog.log in /var/log/syslog.log.1 kopiert und /var/log/syslog.log wird 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. Beispielsweise bleibt /var/log/syslog.log.2014-01-01 bestehen und /var/log/syslog.log.2014-01-02 wird 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 ein Einrichtungsskript für die Installation des CloudWatch-Logs-Agent verwendet haben, können Sie mit /var/awslogs/bin/awslogs-version.sh prüfen, welche Agent-Version Sie verwenden. Dabei werden die Version des Agent und seine großen Abhängigkeiten gedruckt. Wenn Sie den CloudWatch-Logs-Agent mit yum installiert haben, können Sie die Version des CloudWatch-Logs-Agent und Plug-Ins mit "yum info awslogs" und "yum info aws-cli-plugin-cloudwatch-logs" prüfen.

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 das 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 fortgesetzt werden, an der er angehalten wurde, und die Protokolldaten weiterhin übertragen.

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)?

Für den CloudWatch-Logs-Agent sind die Operationen `CreateLogGroup`, `CreateLogStream`, `DescribeLogStreams` und `PutLogEvents` erforderlich. Wenn Sie den neuesten Agent verwendet, wird `DescribeLogStreams` nicht mehr benötigt. Weitere Informationen finden Sie im Beispiel für eine IAM-Richtlinie unten.

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

Ich möchte nicht, dass der CloudWatch-Logs-Agent Protokollgruppen oder Protokoll-Streams automatisch 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`.

Überwachung mit CloudWatch Metriken

CloudWatch Logs sendet CloudWatch jede Minute Metriken an Amazon.

CloudWatch Protokolliert Metriken

Der `AWS/Logs`-Namespace enthält die folgenden Metriken.

Metrik	Beschreibung
`CallCount`	Die Anzahl der angegebenen API-Operationen, die in Ihrem Konto ausgeführt werden. `CallCount` ist eine Metrik zur Nutzung des CloudWatch Logs-Dien stes. Weitere Informationen finden Sie unter CloudWatch Protokolliert Metriken zur Servicenutzung. 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. `AccessDen iedException` `ResourceNotFoundException` Gültige Abmessungen:,,, LogGroupName DestinationType FilterName PolicyLevel Gültige Statistiken: Summe

Metrik	Beschreibung
	Einheiten: keine
`DeliveryThrottling`	Die Anzahl der Protokollereignisse, für die CloudWatch Logs beim Weiterleiten von Daten an das Abonnementziel gedrosselt wurde. 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. `AccessDeniedException` `ResourceNotFoundException` Gültige Abmessungen:,,, LogGroupName DestinationType FilterName PolicyLevel Gültige Statistiken: Summe Einheiten: keine
`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

Metrik	Beschreibung
`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 CloudWatch eingebetteten metrischen Format finden Sie unter Spezifikation: Eingebettetes metrisches Format. Informationen zum Datentyp `MetricDatum` finden Sie MetricDatumin 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
`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 Servicenutzung. Gültige Dimensionen: Klasse, Ressource, Service, Typ Gültige Statistiken: Summe Einheiten: keine

Metrik	Beschreibung
`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
`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

Metrik	Beschreibung
`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
`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 Servicenutzung. Gültige Dimensionen: Klasse, Ressource, Service, Typ Gültige Statistiken: Summe Einheiten: keine

Dimensionen für CloudWatch Logs-Metriken

Die Dimensionen, die Sie mit CloudWatch Logs-Metriken verwenden können, sind in der folgenden Tabelle aufgeführt.

Dimension	Beschreibung
`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 Protokoll gruppe an das Ziel weiterleitet. Der Name des Abonnemen tfilters wird automatisch in ASCII umgewandelt und alle nicht unterstützten Zeichen werden durch ein Fragezeichen (?) ersetzt. CloudWatch

Die Dimensionen für Metriken im Zusammenhang mit Abonnementfiltern auf Kontoebene sind in der folgenden Tabelle aufgeführt.

Dimension	Beschreibung
`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 Protokoll gruppe an das Ziel weiterleitet. Der Name des Abonnemen tfilters wird automatisch in ASCII umgewandelt und alle nicht unterstützten Zeichen werden durch ein Fragezeichen (?) ersetzt. CloudWatch

CloudWatch Protokolliert Metriken zur Servicenutzung

CloudWatch Logs sendet Metriken CloudWatch , um die Nutzung der CloudWatch Logs API-Operationen nachzuverfolgen. 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 unterCloudWatch 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	Beschreibung
`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.

Metrik	Beschreibung
	`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	Beschreibung
`Account metrics`	Verwenden Sie diese Dimension, um eine Summe der Metrik aller CloudWatch Logs-APIs abzurufen. Wenn Sie die Metriken für eine bestimmte API anzeigen möchten, verwenden Sie die anderen Dimensionen, die in dieser Tabelle aufgeführ t sind, und geben Sie den API-Namen als Wert von `Resource` an.
`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 von`None`.
`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-Opera tionsnamen, die in Aktionen aufgelistet sind. Zum Beispiel `PutLogEve nts`

CloudWatch Protokolliert Kontingente

Die folgenden Tabellen enthalten die standardmäßigen Dienstkontingente, auch als Limits bezeichnet, für CloudWatch Logs für ein AWS Konto. Die meisten dieser Service Quotas, aber nicht alle, sind unter dem Amazon CloudWatch Logs-Namespace in der Service Quotas-Konsole aufgeführt. Weitere Informationen zum Beantragen einer Kontingenterhöhung für diese Kontingente finden Sie in dem Verfahren weiter unten in diesem Abschnitt.

Ressource	Standardkontingent
Richtlinien auf Kontoebene	Eine Abonnementfilterrichtlinie auf Kontoebene pro Konto. Eine Datenschutzrichtlinie auf Kontoebene pro Konto. Diese Kontingente können nicht geändert werden.
Detektoren für Anomalien	10 Anomaliedetektoren pro Konto. Dieses Kontingent kann nicht geändert werden.
Batch-Größe	Die maximale Batch-Größe beträgt 1 048 576 Byte. Diese Größe ist die Summe aller Ereignismeldungen in UTF-8 plus 26 Bytes pro Protokollereignis. Dieses Kontingent kann nicht geändert werden.
Datenarchivierung	Bis zu 5 GB kostenlose Datenarchivierung. Dieses Kontingent kann nicht geändert werden.
CreateLogGroup	10 Transaktionen pro Sekunde (TPS/Konto/Region), danach werden Transaktionen gedrosselt. Sie können eine Kontingenterhöhung beantragen.
CreateLogStream	50 Transaktionen pro Sekunde (TPS/Konto/Region), nach denen Transaktionen gedrosselt werden. Sie können eine Kontingenterhöhung beantragen.

Ressource	Standardkontingent
Benutzerdefinierte Datenbezeichner	Jede Datenschutzrichtlinie kann bis zu 10 benutzerd efinierte Datenkennungen enthalten. Sie können eine Kontingenterhöhung beantragen. Jeder reguläre Ausdruck, der eine benutzerdefinierte Daten-ID definiert, kann bis zu 200 Zeichen enthalten. Dieses Kontingent kann nicht geändert werden.
DeleteLogGroup	10 Transaktionen pro Sekunde (TPS/Account/Region), danach werden Transaktionen gedrosselt. Sie können eine Kontingenterhöhung beantragen.
DeleteLogStream	15 Transaktionen pro Sekunde (TPS/Konto/Region), danach werden Transaktionen gedrosselt. Sie können eine Kontingenterhöhung beantragen.
DescribeLogGroups	10 Transaktionen pro Sekunde (TPS/Konto/Region). Sie können eine Kontingenterhöhung beantragen.
DescribeLogStreams	25 Transaktionen pro Sekunde (TPS/Konto/Region). Sie können eine Kontingenterhöhung beantragen.
Erkannte Protokollfelder	CloudWatch Logs Insights kann maximal 1000 Protokoll ereignisfelder in einer Protokollgruppe erkennen. Dieses Kontingent kann nicht geändert werden. Weitere Informationen finden Sie unter Unterstützte Protokolle und erkannte Felder.
Extrahierte Protokollfelder in JSON-Protokollen	CloudWatch Logs Insights kann maximal 200 Protokoll ereignisfelder aus einem JSON-Protokoll extrahieren. Dieses Kontingent kann nicht geändert werden. Weitere Informationen finden Sie unter Unterstützte Protokolle und erkannte Felder.

Ressource	Standardkontingent
Exportaufgabe	Eine aktive (laufende oder ausstehende) Exportaufgabe pro Konto. Dieses Kontingent kann nicht geändert werden.
FilterLogEvents	25 Anfragen pro Sekunde in USA Ost (Nord-Virginia). 5 Anfragen pro Sekunde in den folgenden Regionen: • Asien-Pazifik (Jakarta) • Asien-Pazifik (Osaka) • Europa (Frankfurt) • Kanada West (Calgary) • Israel (Tel Aviv) 10 Anfragen pro Sekunde in anderen Regionen. Dieses Kontingent kann nicht geändert werden.

Ressource	Standardkontingent
GetLogEvents	30 Anfragen pro Sekunde in Europa (Paris). 10 Anforderungen pro Sekunde in den folgenden Regionen: • USA West (Oregon) • Asien-Pazifik (Jakarta) • Asien-Pazifik (Osaka) • Kanada West (Calgary) • Europa (Irland) • Europa (Frankfurt) • Israel (Tel Aviv) 25 Anfragen pro Sekunde in allen anderen Regionen. Dieses Kontingent kann nicht geändert werden. Wir empfehlen Abonnements, wenn Sie fortwährend neue Daten verarbeiten. Wenn Sie historische Daten benötigen , empfiehlt es sich, Ihre Daten in Amazon S3 zu exportier en.
Eingehende Daten	Bis zu 5 GB eingehende Daten, kostenlos. Dieses Kontingent kann nicht geändert werden.
Gleichzeitige Live-Tail-Sitzungen.	15 gleichzeitige Sitzungen Sie können eine Kontingen terhöhung beantragen.
Live Tail: Protokollgruppen, die in einer Sitzung durchsucht werden.	In einer Live-Tail-Sitzung werden höchstens 10 Protokoll gruppen gescannt. Dieses Kontingent kann nicht geändert werden.
Protokollereignisgröße	256 KB (maximal). Dieses Kontingent kann nicht geändert werden.

Ressource	Standardkontingent
Protokollgruppen	1 000 000 Protokollgruppen pro Konto pro Region. Sie können eine Kontingenterhöhung beantragen. Es gibt kein Kontingent dazu, wie viele Protokoll-Streams zu einer Protokollgruppe gehören können.
Metrikfilter	100 pro Protokollgruppe. Dieses Kontingent kann nicht geändert werden.
Metriken im eingebetteten Metrikformat	100 Metriken pro Protokollereignis und 30 Dimensionen pro Metrik. Weitere Informationen zum eingebetteten metrischen Format finden Sie unter Spezifikation: Embedded Metric Format im CloudWatch Amazon-Benutzerhandbuch.
PutLogEvents	Die maximale Batchgröße einer PutLogEvents Anfrage beträgt 1 MB. Diese Größe ist die Summe aller Ereignismeldungen in UTF-8 plus 26 Bytes pro Protokollereignis. 5000 Transaktionen pro Sekunde pro Konto und Region Sie können mithilfe des Dienstes eine Erhöhung des Drosselungskontingents pro Sekunde beantragen. Service Quotas
Zeitüberschreitung für die Abfrageausführung	Bei Abfragen in CloudWatch Logs Insights wird das Timeout nach 60 Minuten überschritten. Dieses Zeitlimit kann nicht geändert werden.
Abgefragte Protokollgruppen	In einer einzigen CloudWatch Logs Insights-Abfrage können maximal 50 Protokollgruppen abgefragt werden. Dieses Kontingent kann nicht geändert werden.

Ressource	Standardkontingent
Gleichzeitigkeit von Abfragen	Für Protokollgruppen der Standardklasse: maximal 30 gleichzeitige CloudWatch Logs Insights-Abfragen, einschließlich Abfragen, die zu Dashboards hinzugefügt wurden. Für Protokollgruppen der Klasse Infrequnt Access maximal 5 gleichzeitige CloudWatch Logs Insights-Abfragen, einschließlich Abfragen, die zu Dashboards hinzugefügt wurden. Diese Kontingente können nicht geändert werden.
Verfügbarkeit abfragen	In der Konsole erstellte Abfragen sind 30 Tage lang über den Befehl Verlauf verfügbar. Dieser Verfügbarkeitszeit raum kann nicht geändert werden. Mithilfe von erstellte Abfragedefinitionen laufen nicht ab. PutQueryDefinition
Verfügbarkeit von Abfrageergebnissen	Ergebnisse aus einer Abfrage können 7 Tage lang abgerufen werden. Dieser Verfügbarkeitszeitraum kann nicht geändert werden.
Anzeige der Abfrageergebnisse in der Konsole	Standardmäßig werden bis zu 1000 Zeilen mit Abfrageergebnissen in der Konsole angezeigt. Sie können den **`limit`**-Befehl in einer Abfrage verwenden, um dies auf bis zu 10.000 Zeilen zu erhöhen. Weitere Informationen finden Sie unter CloudWatch Syntax der Logs Insights-Abfrage.

Ressource	Standardkontingent
Reguläre Ausdrücke	Bis zu 5 Filtermuster mit regulären Ausdrücken für jede Protokollgruppe beim Erstellen von Metrikfiltern oder Abonnementfiltern. Dieses Kontingent kann nicht geändert werden. Bis zu 2 reguläre Ausdrücke für jedes Filtermuster, wenn Sie ein durch Trennzeichen getrenntes oder JSON-Filt ermuster für Metrikfilter und Abonnementfilter oder beim Filtern von Protokollereignissen erstellen.
Ressourcenrichtlinien	Bis zu 10 CloudWatch Protokoll-Ressourcenrichtlinien pro Region und Konto. Dieses Kontingent kann nicht geändert werden.
Gespeicherte Abfragen	Sie können bis zu 1000 CloudWatch Logs Insights-Abfragen pro Region und Konto speichern. Dieses Kontingent kann nicht geändert werden.
Abonnement-Filter	2 pro Protokollgruppe. Dieses Kontingent kann nicht geändert werden.

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 Management Console

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 CLIAWS 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
CloudWatchLogsReadOnlyAccessDie Richtlinie wurde 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 Protokollereigniss	26. November 2023

	en. Weitere Informationen finden Sie unter Musteranalyse.	
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 mit regulären Ausdrücken 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

Datenschutzrichtlinien auf Kontoebene	Sie können jetzt Datenschutzrichtlinien 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

CloudWatchLogsReadOnlyAccessRichtlinie aktualisiert	CloudWatch Protokolliert hinzugefügte Berechtigungen für 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
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.

Änderung	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 Schnittstellen-VPC-Endpunkten.	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

Änderung	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 Protokollieren von Amazon-CloudWatch-Logs-API-Aufrufen 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

Änderung	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.