

Amazon CloudWatch Logs



Amazon CloudWatch Logs: 用户指南

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Amazon Web Services 文档中描述的 Amazon Web Services 服务或功能可能因区域而异。要查看适用于中国区域的差异，请参阅 [中国的 Amazon Web Services 服务入门 \(PDF\)](#)。

Table of Contents

什么是 Amazon CloudWatch Logs ?	1
特征	1
相关 Amazon 服务	2
定价	3
概念	3
账单和成本	4
日志类	5
支持的特征	5
入门	8
前提条件	8
注册 Amazon Web Services 账户	8
保护 IAM 用户	9
设置命令行界面	9
使用统一的 CloudWatch 代理	9
使用之前的 CloudWatch 代理	10
CloudWatch Logs 代理的先决条件	10
快速入门：在运行的 EC2 Linux 实例上安装代理	11
快速入门：启动时在 EC2 Linux 实例上安装代理	17
快速入门：将 CloudWatch Logs 与 Windows Server 2016 实例配合使用	20
快速入门：将 CloudWatch Logs 与 Windows Server 2012 和 Windows Server 2008 实例配合使用	31
报告 CloudWatch Logs 代理的状态	40
启动 CloudWatch Logs 代理	40
停止 CloudWatch Logs 代理	41
CloudWatch Logs 代理参考	41
Amazon CloudFormation 快速入门	52
使用 Amazon SDK	54
使用 CloudWatch Logs Insights 分析日志数据	55
支持的查询语言	57
CloudWatch Logs Insights 查询语言 (Logs Insights QL)	57
OpenSearch 管道处理语言 (PPL)	108
OpenSearch 结构化查询语言 (SQL)	115
使用自然语言生成和更新 CloudWatch Logs Insights 查询	122
示例查询	122

选择不使用您的数据来改善服务	125
支持的日志和发现的字段	125
JSON 日志中的字段	127
创建字段索引以提高查询性能并减少扫描量	129
字段索引语法和配额	131
创建账户级字段索引策略	134
创建日志组级字段索引策略	134
创建查询时的日志组选择选项	135
删除字段索引策略的影响	136
模式分析	136
开始使用模式分析	137
有关 pattern 命令的详细信息	139
保存并重新运行查询	139
将查询添加到控制面板或导出查询结果	141
查看正在运行的查询或查询历史记录	142
使用 Amazon Key Management Service 加密查询结果	143
限制	143
第 1 步：创建 Amazon KMS key	143
步骤 2：设置 KMS 密钥的权限	144
步骤 3：将 KMS 密钥与查询结果关联	146
步骤 4：取消密钥与账户中查询结果的关联	146
根据 CloudWatch Logs Insights 查询结果生成自然语言摘要	146
工作方式	146
区域可用性和数据处理	146
入门	147
权限	148
数据隐私	148
日志异常检测	149
异常和模式的严重性和优先级	150
异常可见性时间	150
抑制异常	150
常见问题	150
在 CloudWatch Logs Insights 中使用异常检测	151
对日志组启用异常检测	152
查看已发现的异常	153
在日志异常检测器上创建警报	155

日志异常检测器发布的指标	157
使用 Amazon KMS 对异常检测器及其结果进行加密	157
限制	157
使用 CloudWatch Logs Live Tail 进行故障排除	162
使用 Amazon CLI SDK 启动 Live Tail 会话	162
print-only	163
交互	163
在控制台中启动 Live Tail 会话	165
跨账户跨区域日志集中化	168
数据集中化概念	168
设置日志集中化	169
先决条件	169
创建集中化规则	169
修改集中化规则	171
查看集中化规则	172
删除集中化规则	172
监控集中化	173
在控制台中监控集中化	173
集中化监控	173
监控规则运行状况	174
使用 Amazon CloudTrail 监控集中化 API 调用	174
使用日志组和日志流	175
创建日志组	175
将日志发送到日志组	175
查看日志数据	176
使用筛选条件模式搜索日志数据	176
使用控制台搜索日志条目	177
使用 Amazon CLI 搜索日志条目	177
从指标定向至日志	178
故障排除	178
更改日志数据保留期	179
标记日志组	180
有关标签的基本知识	180
使用标签跟踪成本	181
标签限制	181
使用 Amazon CLI 标记日志组	182

使用 CloudWatch Logs API 标记日志组	182
使用 Amazon KMS 加密日志数据	182
限制	183
步骤 1：创建 Amazon KMS 密钥	143
步骤 2：设置 KMS 密钥的权限	184
步骤 3：将日志组与 KMS 密钥关联	161
步骤 4：取消日志组与密钥的关联	161
KMS 密钥和加密上下文	189
通过屏蔽帮助保护敏感的日志数据	192
了解数据保护策略	195
创建或使用数据保护策略所需的 IAM 权限	197
创建账户范围的数据保护策略	202
为单个日志组创建数据保护策略	205
查看未屏蔽的数据	208
审计结果报告	208
您可以保护的数据类型	210
在摄取期间转换日志	249
创建和管理日志转换器	250
创建账户级转换器策略	251
编辑或删除账户级转换器策略	252
从头开始创建日志组级日志转换器	252
通过复制现有转换器来创建日志组级转换器	254
编辑日志组级转换器	254
删除日志组级转换器	255
您可以使用的处理器	255
可配置的解析器类型处理器	257
用于 Amazon 公开发布的日志的内置处理器	290
字符串变异处理器	298
JSON 变异处理器	307
数据类型转换器处理器	319
转换指标和错误	322
使用 Amazon OpenSearch Service 进行分析	323
步骤 1：创建与 OpenSearch Service 的集成	324
所需的权限	325
创建集成	332
步骤 2：创建公开发布的日志控制面板	333

查看、编辑或删除公开发布的日志控制面板	334
在 CloudWatch Logs 或 OpenSearch Service 中查看公开发布的日志控制面板	334
向其他 IAM 角色或 IAM 用户授予控制面板查看权限	335
编辑控制面板配置	335
删除公开发布的日志控制面板	335
删除所有与 OpenSearch Service 的公开发布的日志控制面板集成	336
针对用户的 IAM 策略	337
集成所需的权限	338
指标筛选条件	341
概念	342
指标筛选条件的筛选条件模式语法	343
为指标筛选条件配置指标值	344
使用日志事件中的指标发布维度	344
使用录入事件中的值来增加指标的值	347
创建指标筛选条件	348
创建日志组的指标筛选条件	349
示例：对日志事件进行计数	350
示例：对字词的出现次数进行计数	351
示例：对 HTTP 404 代码进行计数	353
示例：对 HTTP 4xx 代码进行计数	355
示例：从 Apache 日志中提取字段并分配维度	356
列出指标筛选条件	358
删除指标筛选条件	359
订阅筛选器	361
概念	362
日志组级别订阅筛选条件	363
示例 1：Kinesis Data Streams 订阅筛选条件	364
示例 2：Amazon Lambda 订阅筛选条件	370
示例 3：Amazon Data Firehose 订阅筛选条件	373
示例 4：Amazon OpenSearch Service 订阅筛选条件	380
账户级订阅筛选条件	380
示例 1：Kinesis Data Streams 订阅筛选条件	381
示例 2：Amazon Lambda 订阅筛选条件	387
示例 3：Amazon Data Firehose 订阅筛选条件	391
跨账户跨区域订阅	398
使用 Kinesis Data Streams 进行跨账户跨区域日志数据共享	399

使用 Firehose 进行跨账户跨区域日志数据共享	417
使用 Kinesis Data Streams 进行跨账户跨区域账户级订阅	431
使用 Firehose 进行跨账户跨区域账户级订阅	447
混淆代理问题防范	459
防止日志递归	460
筛选条件模式语法	462
支持的正则表达式	462
使用正则表达式匹配字词	465
匹配非结构化日志事件中的字词	465
匹配 JSON 日志事件中的字词	469
匹配以空格分隔的日志事件中的字词	477
启用从 Amazon 服务进行日志记录	481
需要额外权限 [V1] 的日志记录	493
发送到 CloudWatch Logs 的日志	494
发送到 Amazon S3 的日志	496
日志发送至 Firehose	500
需要额外权限 [V2] 的日志记录	502
发送到 CloudWatch Logs 的日志	503
发送到 Amazon S3 的日志	506
日志发送至 Firehose	509
跟踪发送到 X-Ray	512
服务特定的权限	515
控制台特定的权限	515
跨账户传输示例	516
创建传输源	517
配置传输到 Amazon S3 存储桶	518
配置向 Firehose 流的传输	521
防止跨服务混淆座席	523
将日志数据导出到 Amazon S3	525
概念	526
使用控制台将日志数据导出到 Amazon S3	527
同账号导出	527
跨账户导出	533
使用 Amazon CLI 将日志数据导出到 Amazon S3	542
同账号导出	542
跨账户导出	549

描述导出任务	557
取消导出任务	559
将数据流式传输到 OpenSearch Service	560
先决条件	560
将日志组订阅到 OpenSearch Service	561
代码示例	563
基本功能	564
操作	564
场景	626
运行大型查询	626
使用计划的事件调用 Lambda 函数	642
安全性	644
数据保护	644
静态加密	645
传输中加密	645
身份和访问管理	646
身份验证	646
访问控制	646
有关管理访问的概述	647
使用基于身份的策略 (IAM 策略)	651
合规性验证	673
恢复能力	673
基础设施安全性	673
接口 VPC 终端节点	674
可用性	674
为 CloudWatch Logs 创建 VPC 终端节点	674
测试 VPC 和 CloudWatch Logs 之间的连接	675
控制对 CloudWatch Logs VPC 终端节点的访问	675
对 VPC 上下文键的支持	676
使用 Amazon CloudTrail 记录 API 和控制台操作	677
CloudTrail 中的查询生成信息	681
了解日志文件条目	682
使用 CloudWatch 指标监控使用情况	684
CloudWatch Logs 指标	684
CloudWatch Logs 指标的维度	688
日志转换器指标和维度	689

CloudWatch Logs 服务使用情况指标	690
服务配额	692
管理您的 CloudWatch Logs 服务配额	698
文档历史记录	700
Amazon 术语表	710

什么是 Amazon CloudWatch Logs ?

您可以使用 Amazon CloudWatch Logs 监控、存储和访问来自 Amazon Elastic Compute Cloud (Amazon EC2) 实例、Amazon CloudTrail、Route 53 和其他来源的日志文件。

CloudWatch Logs 使您能够将所有系统、应用程序和 Amazon 服务中的日志集中在一个高度可扩展的单个服务中。您可以轻松地查看日志、在日志中搜索特定错误代码或模式、根据特定字段筛选日志，或者安全地将这些日志归档以供将来分析。CloudWatch Logs 让您能够按照时间顺序将所有日志（无论它们来自哪里）作为单一且一致的事件流来查看。

此外，CloudWatch Logs 还支持使用强大的查询语言查询日志、审计和屏蔽日志中的敏感数据，以及使用筛选条件或嵌入式日志格式从日志中生成指标。

CloudWatch Logs 支持两个日志类。CloudWatch Logs 标准日志类中的日志组支持所有 CloudWatch Logs 功能。CloudWatch Logs 不频繁访问日志类中的日志组产生较低的提取费用，并且支持一部分标准类功能。有关更多信息，请参阅 [日志类](#)。

特征

- 两种日志类，灵活选择 — CloudWatch Logs 提供两种日志类，以便您可以为不经常访问的日志选择经济高效的选项。对于需要实时监控或其他功能的日志，您还可以选择功能齐全的选项。有关更多信息，请参阅 [日志类](#)。
- 查询您的日志数据 - 您可以使用 CloudWatch Logs Insights 以交互方式搜索和分析您的日志数据。您可以执行查询，以帮助您更高效且有效地响应运算问题。CloudWatch Logs Insights 包括一种专门构建的查询语言以及一些简单但功能强大的命令。我们提供示例查询、命令描述、查询自动完成和日志字段发现功能，可帮助您快速入门。示例查询涉及几种类型的 Amazon 服务日志。要开始使用，请参阅 [使用 CloudWatch Logs Insights 分析日志数据](#)。
- 创建字段索引以提高查询效率 - 您可以为日志事件中的字段创建字段索引。然后，当您在 CloudWatch Logs Insights 查询中使用字段索引时，查询会尝试跳过处理已知不包含索引字段的日志事件。此查询可减少查询的扫描量，从而更快地返回结果。要开始使用，请参阅 [创建字段索引以提高查询性能并减少扫描量](#)。
- 使用 Live Tail 进行检测和调试 — 您可以使用 Live Tail 通过查看提取的新日志事件的流式列表来快速排查事件。您可以近乎实时地查看、筛选和突出显示提取的日志，帮助您快速检测并解决问题。您可以根据指定的术语筛选日志，也可以突出显示包含指定术语的日志，以帮助快速找到所需内容。有关更多信息，请参阅 [使用 CloudWatch Logs Live Tail 进行故障排除](#)。

- 监控来自 Amazon EC2 实例的日志 - 您可以使用 CloudWatch Logs，通过日志数据来监控应用程序和系统。例如，CloudWatch Logs 能够跟踪应用程序日志中发生的错误数，并在错误率超过指定阈值时向您发送通知。CloudWatch Logs 使用您的日志数据进行监控，因此无需更改代码。例如，您可以监控应用程序日志以查找特定的文本字词（如“NullReferenceException”），或计算日志数据中特定位置的文本字词出现的次数（如 Apache 访问日志中的“404”状态代码）。找到您所搜索的字词时，CloudWatch Logs 会向您指定的 CloudWatch 指标报告该数据。日志数据会在传输期间加密，并且会对静态日志数据加密。要开始使用，请参阅[开始使用 CloudWatch Logs](#)。
- 监控 Amazon CloudTrail 记录的事件 - 您可以在 CloudWatch 中创建告警并接收 CloudTrail 捕获的特定 API 活动的通知，然后使用通知执行故障排除。要了解其用法，请参阅 Amazon CloudTrail 用户指南中的[将 CloudTrail 事件发送到 CloudWatch Logs](#)。
- 审计和屏蔽敏感数据 - 如果您的日志中包含敏感数据，可以利用数据保护策略来帮助保护敏感数据。这些策略可让您审核并屏蔽敏感数据。如果您启用数据保护，则默认情况下，将屏蔽与您所选数据标识符相匹配的敏感数据。有关更多信息，请参阅[通过屏蔽帮助保护敏感的日志数据](#)。
- 日志保留期 - 默认情况下，日志将无限期保留且永不过期。您可以调整每个日志组的保留策略，保持无限期保留或选择介于 10 年和一天之间的保留期。
- 归档日志数据 - 您可以使用 CloudWatch Logs 在高持久性存储中存储日志数据。CloudWatch Logs 代理支持您轻松快速地将已轮换和未轮换的日志文件从主机移动到日志服务。然后，您可以按需访问原始日志数据。
- 记录 Route 53 DNS 查询 - 您可以使用 CloudWatch Logs 记录有关 Route 53 接收的 DNS 查询的信息。有关更多信息，请参阅 Amazon Route 53 开发人员指南中的[记录 DNS 查询](#)。
- 跨账户和区域集中日志 - 您可以使用 CloudWatch Logs 集中化来定义跨账户和跨区域集中化规则，以便将跨多个账户和区域环境摄取的日志数据复制到中央区域和账户中。您可以配置中央账户内的备份区域以提高恢复能力，为中央账户中新创建的日志组定义加密设置，并针对来自特定源账户和区域的日志事件创建指标和订阅筛选条件，从而增强筛选能力。

相关 Amazon 服务

以下服务可与 CloudWatch Logs 一起使用：

- Amazon CloudTrail 是一项 Web 服务，可用来监控对您的账户的 CloudWatch Logs API 进行的调用，包括由 Amazon Web Services 管理控制台、Amazon Command Line Interface (Amazon CLI) 和其他服务发出的调用。启用 CloudTrail 日志记录后，CloudTrail 将捕获您的账户中的 API 调用，并将日志文件传输到指定的 Amazon S3 存储桶。每个日志文件可以包含一个或多个记录，具体取决于为满足某个请求而必须执行的操作的数量。有关 Amazon CloudTrail 的更多信息，请参阅《Amazon CloudTrail 用户指南》中的[什么是 Amazon CloudTrail？](#)有关由 CloudWatch 写入

CloudTrail 日志文件的数据类型的示例，请参阅 [在 Amazon CloudTrail 中记录 CloudWatch Logs API 和控制台操作](#)。

- Amazon Identity and Access Management (IAM) 是一种 Web 服务，可帮助您安全地控制用户对 Amazon 资源的访问权限。使用 IAM 可以控制可使用您的 Amazon 资源（身份验证）的人员、他们可以使用资源以及使用这些资源的方式（授权）。有关更多信息，请参阅 IAM 用户指南中的 [什么是 IAM ?](#)。
- Amazon Kinesis Data Streams 是一项 Web 服务，可用于实现快速而持续的数据接收和聚合。使用的数据类型包括 IT 基础设施日志数据、应用程序日志、社交媒体、市场数据源和 Web 点击流数据。由于数据引入和处理的响应时间是实时的，因此处理通常是轻量级的。有关更多信息，请参阅 Amazon Kinesis Data Streams 开发人员指南 中的 [什么是 Amazon Kinesis Data Streams ?](#)。
- Amazon Lambda 是一项 Web 服务，可用于轻松地构建快速响应新信息的应用程序。将您的应用程序代码作为 Lambda 函数上传，Lambda 会在高可用性计算基础设施上运行您的代码，执行计算资源的所有管理工作，包括服务器和操作系统维护、容量预配和弹性伸缩、代码和安全补丁部署以及代码监控和日志记录。您只需要以 Lambda 支持的一种语言提供您的代码。有关更多信息，请参阅 Amazon Lambda 开发人员指南 中的 [什么是 Amazon Lambda ?](#)。

定价

注册 Amazon 后，您可以通过 [Amazon 免费套餐](#) 开始免费使用 CloudWatch Logs。

标准费率适用于由其他服务使用 CloudWatch Logs 存储的日志（例如，Amazon VPC 流日志和 Lambda 日志）。

有关定价的更多信息，请参阅 [Amazon CloudWatch 定价](#)。

有关如何分析 CloudWatch Logs 和 CloudWatch 的成本和使用情况的更多信息，以及有关如何降低成本的最佳实践，请参阅 [CloudWatch 账单和成本](#)。

Amazon CloudWatch Logs 的概念

下面介绍便于您了解和使用 CloudWatch Logs 的核心术语和概念。

日志类

CloudWatch Logs 提供两类日志组。标准日志类是适用于需要实时监控的日志或您经常访问的日志的功能齐全的选项。对于不经常访问的日志，不频繁访问日志类是一种低成本选项。它支持一部分标准日志类功能。

日志事件

日志事件是对受监视的应用程序或资源记录的一些活动的记录。CloudWatch Logs 理解的日志事件记录包含两个属性：事件发生时的时间戳和原始事件消息。事件消息必须采用 UTF-8 编码。

日志流

日志流是共享同一来源的一系列日志事件。具体而言，日志流通常用于表示来自正在监控之下的应用程序实例或资源的事件序列。例如，日志流可能与特定主机上的 Apache 访问日志相关联。当您不再需要日志流时，可使用 [aws logs delete-log-stream](#) 命令将其删除。

日志组

日志组定义日志保留期、监控和访问控制设置都相同的日志流组。每个日志流必须属于一个日志组。例如，如果每个主机上的 Apache 访问日志都有一个单独的日志流，您可以将这些日志流分到一个名为 `MyWebsite.com/Apache/access_log` 的单独日志组。

对可属于一个日志组的日志流数没有限制。

指标筛选条件

您可以使用指标筛选条件从已接收的事件中提取指标观察数据并将它们转换为 CloudWatch 指标中的数据点。指标筛选条件将分配给日志组，分配给日志组的所有筛选条件都将应用于其日志流。

保留期设置

保留期设置可用于指定将日志事件保留在 CloudWatch Logs 中多长时间。将会自动删除过期的日志事件。和指标筛选器一样，保留期设置也会分配给日志组，分配给日志组的保留期将应用于其日志流。

Amazon CloudWatch Logs 账单和成本

有关如何分析 CloudWatch Logs 和 CloudWatch 的成本和使用情况的详细信息，以及有关如何降低成本的最佳实践，请参阅 [CloudWatch 账单和成本](#)。

有关定价的更多信息，请参阅 [Amazon CloudWatch 定价](#)。

注册 Amazon 后，您可以通过 [Amazon 免费套餐](#) 开始免费使用 CloudWatch Logs。

标准费率适用于由其他服务使用 CloudWatch Logs 存储的日志（例如，Amazon VPC 流日志和 Lambda 日志）。

日志类

CloudWatch Logs 提供两类日志组：

- CloudWatch Logs 标准日志类是适用于需要实时监控的日志或您经常访问的日志的功能齐全的选项。
- CloudWatch Logs 不频繁访问日志类是一个新的日志类，您可以使用它来经济高效地整合日志。此日志类提供了 CloudWatch Logs 功能的子集，包括托管提取、存储、跨账户日志分析和加密，并且每 GB 的提取价格较低。不频繁访问日志类非常适合对不频繁访问的日志进行临时查询和事后取证分析。

 Note

对于费用，标准和频繁访问日志类仅在提取成本方面有所不同。每个日志类的存储费用和 CloudWatch Logs Insights 费用相同。

有关 CloudWatch Logs 定价的更多信息，请参阅 [Amazon CloudWatch 定价](#)。

 Important

在创建日志组后，无法更改其日志类。

支持的特征

下表列出了每个日志类的功能。

功能	Standard	不频繁访问
完全托管日志提取和存储	是 ✓	是 ✓
跨账户功能	是 ✓	是 ✓
使用 Amazon KMS 进行加密	是 ✓	是 ✓

功能	Standard	不频繁访问	
CloudWatch Logs Insights 查询命令	是 ✓	是 ✓ (大多数命令 – 请参阅 日志类中支持的 Logs Insights QL 命令 。)	
CloudWatch Logs Insights 发现的字段	是 ✓	是 ✓	
使用 OpenSearch PPL 或 OpenSearch SQL 在 CloudWatch Logs Insights 中进行查询；	是 ✓	否	
自然语言查询助手	是 ✓	否	
CloudWatch Logs 异常检测	是 ✓	否	
Live Tail	是 ✓	否	
字段索引	是 ✓	否	
与之前的时间范围比较	是 ✓	否	
订阅筛选条件	是 ✓	否	
导出到 Amazon S3	是 ✓	否	
GetLogEvents 和 FilterLogEvents API 操作	是 ✓	不支持。使用 CloudWatch Logs Insights 查看存储在 不频繁访问日志类中的日志组中的日志事件。	
指标筛选条件	是 ✓	否	
Container Insights 日志提取	是 ✓	否	

功能	Standard	不频繁访问	
Lambda Insights 日志提取	是 ✓	否	
通过屏蔽保护敏感数据	是 ✓	否	
嵌入式指标格式	是 ✓	否	

 Note

除了这两个日志类之外，还有一个Delivery日志类。仅将Delivery日志类用于传送要存储在 Amazon S3 或 Amazon Data Firehose 中的 Amazon Lambda 日志。传输类的日志组中的日志事件在 CloudWatch Logs 中仅保留一天。此日志类不提供丰富的 CloudWatch Logs 功能，例如 CloudWatch Logs Insights 查询。

开始使用 CloudWatch Logs

要将 Amazon EC2 实例和本地服务器中的日志收集到 CloudWatch Logs，请使用统一 CloudWatch 代理。它让您能够使用一个代理收集日志和高级指标。它跨操作系统提供支持，包括运行 Windows Server 的服务器。该代理还提供更好的性能。

如果使用统一 CloudWatch 代理收集 CloudWatch 指标，则它能收集其他系统指标，获得来宾可见性。它还支持使用 StatsD 或 collectd 收集自定义指标。

有关更多信息，请参阅 Amazon CloudWatch 用户指南中的[安装 CloudWatch 代理](#)文档。

较旧的 CloudWatch Logs 代理仅支持从运行 Linux 的服务器中收集日志，现已弃用，且不再受支持。有关从较旧 CloudWatch Logs 代理迁移到统一代理的信息，请参阅[通过向导创建 CloudWatch 代理配置文件](#)。

目录

- [前提条件](#)
- [使用统一的 CloudWatch 代理开始使用 CloudWatch Logs](#)
- [使用之前的 CloudWatch 代理以开始使用 CloudWatch Logs](#)
- [快速入门：使用 Amazon CloudFormation 来开始使用 CloudWatch Logs](#)

前提条件

要使用 Amazon CloudWatch Logs，您需要一个 Amazon 账户。利用您的 Amazon 账户，可以使用服务（例如 Amazon EC2）生成可在 CloudWatch 控制台（一种基于 Web 的界面）中查看的日志。此外，您还可以安装和配置 Amazon Command Line Interface (Amazon CLI)。

注册 Amazon Web Services 账户

如果您还没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

注册 Amazon Web Services 账户

1. 打开 <https://portal.aws.amazon.com/billing/signup>。
2. 按照屏幕上的说明操作。

在注册时，将接到电话或收到短信，要求使用电话键盘输入一个验证码。

当您注册 Amazon Web Services 账户时，系统将会创建一个 Amazon Web Services 账户根用户。根用户有权访问该账户中的所有 Amazon Web Services 服务和资源。作为最佳安全实践，请为用户分配管理访问权限，并且只使用根用户来执行[需要根用户访问权限的任务](#)。

注册过程完成后，Amazon 会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 <https://aws.amazon.com/> 并选择我的账户来查看当前的账户活动并管理您的账户。

保护 IAM 用户

注册 Amazon Web Services 账户后，启用多重身份验证 (MFA) 保护您的管理用户。有关说明，请参阅《IAM 用户指南》中的[为 IAM 用户启用虚拟 MFA 设备 \(控制台\)](#)。

要授予其他用户访问您的 Amazon Web Services 账户资源的权限，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA 并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅《IAM 用户指南》中的以下主题：

- [在您的 Amazon Web Services 账户中创建 IAM 用户](#)
- [适用于 Amazon 资源的访问权限管理](#)
- [基于 IAM 身份的策略示例](#)

设置命令行界面

您可以使用 Amazon CLI 来执行 CloudWatch Logs 操作。

有关如何安装和配置 Amazon CLI 的信息，请参阅 Amazon Command Line Interface 用户指南中的[设置 Amazon Command Line Interface](#)。

使用统一的 CloudWatch 代理开始使用 CloudWatch Logs

有关使用统一的 CloudWatch Logs 代理的更多信息，请参阅 Amazon CloudWatch 用户指南中的[使用 CloudWatch 代理从 Amazon EC2 实例和本地服务器中收集指标和日志](#)。完成本部分中列出的步骤以安装、配置和启动代理。如果不使用此代理收集 CloudWatch 指标，则可以忽略有关指标的所有部分。

如果您正在使用较旧的 CloudWatch Logs 代理并打算迁移到新的统一代理，建议您使用新代理程序包中包含的向导。此向导能够读取当前的 CloudWatch Logs 代理配置文件，并设置 CloudWatch 代理来

收集相同的日志。有关此向导的更多信息，请参阅 Amazon CloudWatch 用户指南中的[使用向导创建 CloudWatch 代理配置文件](#)。

使用之前的 CloudWatch 代理以开始使用 CloudWatch Logs

Important

CloudWatch 包含一个可收集来自 EC2 实例和本地服务器的日志和指标的统一定 CloudWatch 代理。仅限日志的较旧代理现已弃用，且不再受支持。

有关从仅限日志的较旧代理迁移到统一定代理的信息，请参阅[通过向导创建 CloudWatch 代理配置文件](#)。

本节的其余部分将为仍在使用的客户介绍较早版本的 CloudWatch Logs 代理的使用。

使用 CloudWatch Logs 代理，您可以发布来自运行 Linux 或 Windows Server 的 Amazon EC2 实例的日志数据和来自 Amazon CloudTrail 的已录入事件。建议您改用 CloudWatch 统一定代理发布日志数据。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的[使用 CloudWatch 代理从 Amazon EC2 实例和本地服务器中收集指标和日志](#)。

目录

- [CloudWatch Logs 代理的先决条件](#)
- [快速入门：在运行的 EC2 Linux 实例上安装和配置 CloudWatch Logs 代理](#)
- [快速入门：启动时在 EC2 Linux 实例上安装和配置 CloudWatch Logs 代理](#)
- [快速入门：让运行 Windows Server 2016 的 Amazon EC2 实例使用 CloudWatch Logs 代理将日志发送到 CloudWatch Logs](#)
- [快速入门：让您的运行 Windows Server 2012 和 Windows Server 2008 的 Amazon EC2 实例将日志发送到 CloudWatch Logs](#)
- [报告 CloudWatch Logs 代理的状态](#)
- [启动 CloudWatch Logs 代理](#)
- [停止 CloudWatch Logs 代理](#)
- [CloudWatch Logs 代理参考](#)

CloudWatch Logs 代理的先决条件

CloudWatch Logs 代理需要 Python 版本 2.7、3.0 或 3.3，以及以下任意版本的 Linux：

- Amazon Linux 2014.03.02 或更高版本。不支持 Amazon Linux 2
- Ubuntu Server 版本 12.04、14.04 或 16.04
- CentOS 版本 6、6.3、6.4、6.5 或 7.0
- Red Hat Enterprise Linux (RHEL) 版本 6.5 或 7.0
- Debian 8.0

快速入门：在运行的 EC2 Linux 实例上安装和配置 CloudWatch Logs 代理

Important

较早版本的日志代理已弃用。CloudWatch 包含一个可收集来自 EC2 实例和本地服务器的日志和指标的统一代理。有关更多信息，请参阅 [开始使用 CloudWatch Logs](#)。

有关从较旧 CloudWatch Logs 代理迁移到统一代理的信息，请参阅[通过向导创建 CloudWatch 代理配置文件](#)。

较早版本的日志代理仅支持 Python 2.6 至 3.5 版本。此外，较早版本的 CloudWatch Logs 代理不支持实例元数据服务版本 2 (IMDSv2)。如果您的服务器使用 IMDSv2，则必须使用较新版本的统一代理，而不是较早版本的 CloudWatch Logs 代理。

本节的其余部分将为仍在使用的客户介绍较早版本的 CloudWatch Logs 代理的使用。

Tip

CloudWatch 包含一个可收集来自 EC2 实例和本地服务器的日志和指标的新统一代理。如果您尚未使用较旧的 CloudWatch Logs 代理，我们建议您使用较新的统一 CloudWatch 代理。有关更多信息，请参阅 [开始使用 CloudWatch Logs](#)。

此外，较早版本的代理不支持实例元数据服务版本 2 (IMDSv2)。如果您的服务器使用 IMDSv2，则必须使用较新版本的统一代理，而不是较早版本的 CloudWatch Logs 代理。本节的其余部分将介绍较旧的 CloudWatch Logs 代理的使用。

在运行的 EC2 Linux 实例上配置较旧的 CloudWatch Logs 代理

您可以使用 CloudWatch Logs 代理安装程序在现有 EC2 实例中安装和配置 CloudWatch Logs 代理。安装完成后，日志自动从实例流向您在安装代理时创建的日志流。代理会确认它已启动，并保持运行状态，直到您禁用它为止。

除了该代理之外，您还可以使用 Amazon CLI、CloudWatch Logs 开发工具包或 CloudWatch Logs API 发布日志数据。Amazon CLI 非常适合用于通过命令行或脚本发布数据。CloudWatch Logs 开发工具包非常适合用于直接从应用程序发布日志数据或构建您自己的日志发布应用程序。

步骤 1：为 CloudWatch Logs 配置 IAM 角色或用户

CloudWatch Logs 代理支持 IAM 角色和用户。如果您的实例已有一个与之关联的 IAM 角色，请确保包含下面的 IAM policy。如果您尚未将 IAM 角色分配给实例，则可以在后续步骤中使用 IAM 凭证，也可以向该实例分配 IAM 角色。有关更多信息，请参阅[将 IAM 角色附加到实例](#)。

为 CloudWatch Logs 配置 IAM 角色或用户

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Roles (角色)。
3. 通过选择角色名称来选择角色 (不要选中名称旁边的复选框)。
4. 依次选择 Attach Policies (附加策略) 和 Create Policy (创建策略)。

打开一个新的浏览器选项卡或窗口。

5. 选择 JSON 选项卡，然后键入以下 JSON 策略文档中的文本。

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

6. 完成后，选择 Review policy (查看策略)。策略验证程序将报告任何语法错误。

7. 在 Review Policy (查看策略) 页上, 为创建的策略键入 Name (名称) 和 Description (说明) (可选)。查看策略 Summary (摘要) 以查看您的策略授予的权限。然后, 选择 Create policy (创建策略) 以保存您的工作。
8. 关闭浏览器选项卡或窗口, 然后返回到您角色的 Add permissions (添加权限) 页面。选择 Refresh (刷新), 然后选择新策略以将其附加到您的组。
9. 选择 Attach Policy (附加策略)。

步骤 2: 在现有 Amazon EC2 实例上安装和配置 CloudWatch Logs

安装 CloudWatch Logs 代理的过程取决于 Amazon EC2 实例运行的是 Amazon Linux、Ubuntu、CentOS 还是 Red Hat。请根据实例上的 Linux 版本采用适当的步骤。

在现有 Amazon Linux 实例上安装和配置 CloudWatch Logs

从 Amazon Linux AMI 2014.09 开始, CloudWatch Logs 代理作为 RPM 安装随 awslogs 软件包提供。早期版本的 Amazon Linux 可以通过使用 `sudo yum update -y` 命令更新其实例来访问 awslogs 软件包。通过将 awslogs 软件包作为 RPM 安装 (而不是使用 CloudWatch Logs 安装程序), 实例可从 Amazon 接收定期软件包更新和补丁, 而不必手动重新安装 CloudWatch Logs 代理。

Warning

如果以前是使用 Python 脚本安装的 CloudWatch Logs 代理, 请勿使用 RPM 安装方法更新该代理。这样做可能会导致配置问题, 使得 CloudWatch Logs 代理无法将您的日志发送到 CloudWatch。

1. 连接到 Amazon Linux 实例。有关详细信息, 请参阅《Amazon EC2 用户指南》中的[连接到您的实例](#)。

有关连接问题的更多信息, 请参阅《Amazon EC2 用户指南》中的[排查实例的连接问题](#)。

2. 更新您的 Amazon Linux 实例以在软件包存储库中选取最新更改。

```
sudo yum update -y
```

3. 安装 awslogs 软件包。这是在 Amazon Linux 实例上安装 awslogs 的推荐方法。

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

4. 编辑 `/etc/awslogs/awslogs.conf` 文件以配置要跟踪的日志。有关编辑此文件的更多信息，请参阅 [CloudWatch Logs 代理参考](#)。
5. 默认情况下，`/etc/awslogs/awscli.conf` 指向 `us-east-1` 区域。要将日志推送到其他区域，请编辑 `awscli.conf` 文件并指定该区域。
6. 启动 `awslogs` 服务。

```
sudo service awslogs start
```

如果您运行的是 Amazon Linux 2，请使用以下命令启动 `awslogs`。

```
sudo systemctl start awslogsd
```

7. (可选) 检查 `/var/log/awslogs.log` 文件中是否有在启动服务时记录的错误。
8. (可选) 在每次系统启动时运行以下命令以启动 `awslogs` 服务。

```
sudo chkconfig awslogs on
```

如果您运行的是 Amazon Linux 2，请在每次系统启动时使用以下命令启动该服务。

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

9. 在代理运行一段时间后，可以在 CloudWatch 控制台看到新创建的日志组和日志流。

有关更多信息，请参阅 [查看发送到 CloudWatch Logs 的日志数据](#)。

在现有 Ubuntu Server、CentOS 或 Red Hat 实例上安装和配置 CloudWatch Logs

如果使用运行 Ubuntu Server、CentOS 或 Red Hat 的 AMI，请使用以下过程在实例上手动安装 CloudWatch Logs 代理。

1. 连接到 EC2 实例。有关详细信息，请参阅《Amazon EC2 用户指南》中的 [连接到您的实例](#)。

有关连接问题的更多信息，请参阅《Amazon EC2 用户指南》中的 [排查实例的连接问题](#)。

2. 使用两个选项之一运行 CloudWatch Logs 代理安装程序。您可以直接从 Internet 运行，也可以下载文件并独立运行。

 Note

如果运行的是 CentOS 6.x、Red Hat 6.x 或 Ubuntu 12.04，请使用相应的步骤下载和运行单独的安装程序。这些系统不支持直接从 Internet 安装 CloudWatch Logs 代理。

 Note

在 Ubuntu 中，在运行以下命令之前运行 `apt-get update`。

要直接从 Internet 运行，请使用以下命令并按照提示操作：

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

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

如果上述命令不起作用，请尝试以下命令：

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

要下载并独立运行它，请使用以下命令并按照提示操作：

```
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
```

您可以通过指定 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 或 sa-east-1 区域来安装 CloudWatch Logs 代理。

 Note

有关 `awslogs-agent-setup` 的当前版本和版本历史记录的更多信息，请参阅 [CHANGELOG.txt](#)。

在设置期间，CloudWatch Logs 代理安装程序需要特定信息。在开始之前，您需要知道要监视的日志文件及其时间戳格式。您应准备好以下信息。

项目	描述
Amazon访问密钥 ID	如果使用 IAM 角色，请按 Enter。否则，请输入您的Amazon访问密钥 ID。
Amazon秘密访问密钥	如果使用 IAM 角色，请按 Enter。否则，请输入您的Amazon秘密访问密钥。
默认区域名称	按 Enter。默认区域为 us-east-2。您可以将它设置为 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 或 sa-east-1。
默认输出格式	保留空白并按 Enter。
要上传的日志文件的路径	包含待发送日志数据的文件的位置。安装程序将为您提供路径建议。
目标日志组名称	日志组的名称。安装程序将为您提供日志组名称建议。
目标日志流名称	默认情况下，这是主机名称。安装程序将为您提供主机名称建议。
时间戳格式	指定在指定日志文件中使用的的时间戳的格式。选择自定义可指定您自己的格式。

项目	描述
初始位置	数据的上传方式。设置为 start_of_file 将上传数据文件中的所有内容。设置为 end_of_file 将仅上传最近追加的数据。

完成这些步骤后，安装程序会询问您是否需要配置另一个日志文件。对于每个日志文件，您都可以运行任意次此过程。如果没有其他要监控的日志文件，当安装程序提示设置其他日志时，请选择 N（否）。有关代理配置文件中的设置的更多信息，请参阅 [CloudWatch Logs 代理参考](#)。

 Note

不支持配置多个日志源将数据发送到单个日志流。

- 在代理运行一段时间后，可以在 CloudWatch 控制台看到新创建的日志组和日志流。

有关更多信息，请参阅 [查看发送到 CloudWatch Logs 的日志数据](#)。

快速入门：启动时在 EC2 Linux 实例上安装和配置 CloudWatch Logs 代理

 Tip

本节中讨论的较旧 CloudWatch Logs 代理即将弃用。强烈建议您改为使用可以收集日志和指标的新统一 CloudWatch 代理。此外，较旧 CloudWatch Logs 代理需要 Python 3.3 或更早版本，默认情况下，新 EC2 实例上未安装这些版本。有关统一 CloudWatch 代理的更多信息，请参阅[安装 CloudWatch 代理](#)。

本节的其余部分将介绍较旧的 CloudWatch Logs 代理的使用。

启动时在 EC2 Linux 实例上安装较旧的 CloudWatch Logs 代理

您可以使用 Amazon EC2 用户数据，这是 Amazon EC2 的一项功能，允许在启动时将参数信息传递给实例，以在该实例中安装和配置 CloudWatch Logs 代理。要将 CloudWatch Logs 代理的安装和配置信息传递给 Amazon EC2，您可以提供处于某个网络位置（例如 Amazon S3 存储桶）的配置文件。

不支持配置多个日志源将数据发送到单个日志流。

先决条件

创建用于描述所有日志组和日志流的代理配置文件。这是一个文本文件，描述要监控的日志文件以及要将这些文件上传到的日志组和日志流。代理使用此配置文件并开始监控并上传其中描述的所有日志文件。有关代理配置文件中的设置的更多信息，请参阅 [CloudWatch Logs 代理参考](#)。

以下是适用于 Amazon Linux 2 的一个示例代理配置文件

```
[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
```

以下是适用于 Ubuntu 的一个示例代理配置文件

```
[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
```

配置您的 IAM 角色

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略) 和 Create Policy (创建策略)。
3. 在 Create Policy (创建策略) 页面上，对 Create Your Own Policy (创建您自己的策略) 选择 Select (选择)。有关创建自定义策略的更多信息，请参阅《Amazon EC2 用户指南》中的 [Amazon EC2 的 IAM 策略](#)。
4. 在 Review Policy (查看策略) 页面上，为 Policy Name (策略名称) 键入策略的名称。
5. 对于 Policy Document (策略文档)，粘贴以下策略：

JSON

```
{
```

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

6. 选择 Create Policy (创建策略)。
7. 在导航窗格中，选择 Roles (角色) 和 Create New Role (创建新角色)。
8. 在 Set Role Name (设置角色名称) 页面上，请键入角色名称，然后选择 Next Step (下一步)。
9. 在 Select Role Type (选择角色类型) 部分，选择 Amazon EC2 旁的 Select (选择)。
10. 在 Attach Policy (附加策略) 页面的表标题中，依次选择 Policy Type (策略类型) 和 Customer Managed (客户托管)。
11. 选择您创建的 IAM policy，然后选择 Next Step (下一步)。
12. 选择创建角色。

有关用户和策略的更多信息，请参阅 IAM 用户指南中的 [IAM 用户和组](#) 以及 [管理 IAM policy](#)。

启动新实例和启用 CloudWatch Logs

1. 通过以下网址打开 Amazon EC2 控制台：<https://console.aws.amazon.com/ec2/>。

2. 选择 Launch Instance (启动实例) 。

有关更多信息，请参阅《Amazon EC2 用户指南》中的[启动实例](#)。

- 在 Step 1: Choose an Amazon Machine Image (AMI) (步骤 1：选择 Amazon Machine Image (AMI)) 页面上，选择要启动的 Linux 实例类型，然后在 Step 2: Choose an Instance Type (步骤 2：选择实例类型) 页面上，选择 Next: Configure Instance Details (下一步：配置实例详细信息) 。

确保 Amazon Machine Image (AMI) 中包含了 [cloud-init](#)。Amazon Linux AMI 及适用于 Ubuntu 和 RHEL 的 AMI 已包含 cloud-init，但 Amazon Web Services Marketplace 中的 CentOS 和其他 AMI 可能未包含 cloud-init。

- 在 Step 3: Configure Instance Details (步骤 3：配置实例详细信息) 页面上，为 IAM role (IAM 角色) 选择您创建的 IAM 角色。
- 在 Advanced Details (高级详细信息) 下，将以下脚本粘贴到 User data (用户数据) 的框中。然后，通过将 -c 选项的值更改为您的代理配置文件的位置来更新该脚本：

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

- 对实例进行任何其他更改，检查启动设置，然后选择 Launch (启动) 。
- 在代理运行一段时间后，可以在 CloudWatch 控制台看到新创建的日志组和日志流。

有关更多信息，请参阅 [查看发送到 CloudWatch Logs 的日志数据](#)。

快速入门：让运行 Windows Server 2016 的 Amazon EC2 实例使用 CloudWatch Logs 代理将日志发送到 CloudWatch Logs

Tip

CloudWatch 包含一个可收集来自 EC2 实例和本地服务器的日志和指标的新统一代理。我们建议您使用较新的统一 CloudWatch 代理。有关更多信息，请参阅 [开始使用 CloudWatch Logs](#)。本节的其余部分将介绍较旧的 CloudWatch Logs 代理的使用。

使运行 Windows Server 2016 的 Amazon EC2 实例能够使用较旧的 CloudWatch Logs 代理将日志发送到 CloudWatch Logs

您可以使用多种方法让运行 Windows Server 2016 的实例将日志发送到 CloudWatch Logs。此部分中的步骤使用 Systems Manager Run Command。有关其他可能方法的信息，请参阅[向 Amazon CloudWatch 发送日志、事件和性能计数器](#)。

步骤

- [下载示例配置文件](#)
- [为 CloudWatch 配置 JSON 文件](#)
- [为 Systems Manager 创建 IAM 角色](#)
- [验证 Systems Manager 的先决条件](#)
- [验证 Internet 访问权限](#)
- [使用 Systems Manager Run Command 启用 CloudWatch Logs](#)

下载示例配置文件

将以下示例文件下载到您的计算机：[AWS.EC2.Windows.CloudWatch.json](#)。

为 CloudWatch 配置 JSON 文件

您通过在配置文件中指定选项来确定要发送到 CloudWatch 的日志。创建此文件并指定您的选择的过程可能需要 30 分钟或更长时间才能完成。完成此任务一次后，您可以在所有实例上重复使用此配置文件。

步骤

- [步骤 1：启用 CloudWatch Logs](#)
- [步骤 2：为 CloudWatch 配置设置](#)
- [步骤 3：配置要发送的数据](#)
- [步骤 4：配置流程控制](#)
- [步骤 5：保存 JSON 内容](#)

步骤 1：启用 CloudWatch Logs

在 JSON 文件的顶部，将 IsEnabled 的“false”改为“true”：

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

步骤 2：为 CloudWatch 配置设置

指定凭证、区域、日志组名称和日志流命名空间。这使实例能够将日志数据发送到 CloudWatch Logs。要将同一日志数据发送到其他位置，您可以添加包含唯一 ID 的额外部分（例如，“CloudWatchLogs2”和“CloudWatchLogs3”）并为每个 ID 添加一个不同的区域。

配置设置以将日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 CloudWatchLogs 部分。

```
{
  "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. 将 AccessKey 和 SecretKey 字段留空。您将使用 IAM 角色配置凭证。
3. 对于 Region，键入要向其中发送日志数据的区域（例如 us-east-2）。
4. 对于 LogGroup，键入您的日志组的名称。此名称显示在 CloudWatch 控制台的 Log Groups（日志组）屏幕上。
5. 对于 LogStream，键入目标日志流。此名称显示在 CloudWatch 控制台的 Log Groups（日志组）> Streams（流）屏幕上。

如果使用 {instance_id}，则在默认情况下，日志流名称是该实例的实例 ID。

如果指定的日志流名称不存在，则 CloudWatch Logs 会自动创建该名称。您可以使用文字字符串、预定义变量 {instance_id}、{hostname} 和 {ip_address} 或它们的组合定义日志流名称。

步骤 3：配置要发送的数据

您可以将事件日志数据、Windows 事件跟踪 (ETW) 数据和其他日志数据发送到 CloudWatch Logs。

将 Windows 应用程序事件日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 ApplicationEventLog 部分。

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

2. 对于 Levels，指定要上传的消息类型。可以指定以下值之一：

- **1** – 仅上传错误消息。
- **2** – 仅上传警告消息。
- **4** – 仅上传信息消息。

您可以将这些值组合在一起，以包含多种类型的消息。例如，一个包含 **3** 个上传错误消息 (**1**) 和警告消息 (**2**) 的值。一个包含 **7** 个上传错误消息 (**1**)、警告消息 (**2**) 和信息消息 (**4**) 的值。

将安全日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 SecurityEventLog 部分。

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

2. 对于 Levels，键入 7 以上传所有消息。

将系统事件日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 SystemEventLog 部分。

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

2. 对于 Levels，指定要上传的消息类型。可以指定以下值之一：

- 1 – 仅上传错误消息。
- 2 – 仅上传警告消息。
- 4 – 仅上传信息消息。

您可以将这些值组合在一起，以包含多种类型的消息。例如，一个包含 3 个上传错误消息 (1) 和警告消息 (2) 的值。一个包含 7 个上传错误消息 (1)、警告消息 (2) 和信息消息 (4) 的值。

将其他类型的事件日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，添加新部分。每个部分必须有一个独立的 Id。

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

2. 对于 Id，键入要上传的日志的名称（例如 **WindowsBackup**）。

3. 对于 LogName，请键入要上传的日志的名称。您可以按如下步骤找到日志的名称。
 - a. 打开事件查看器。
 - b. 在导航窗格中，选择 Applications and Services Logs（应用程序和服务日志）。
 - c. 导航到该日志，然后选择 Actions（操作）、Properties（属性）。
4. 对于 Levels，指定要上传的消息类型。可以指定以下值之一：
 - **1** – 仅上传错误消息。
 - **2** – 仅上传警告消息。
 - **4** – 仅上传信息消息。

您可以将这些值组合在一起，以包含多种类型的消息。例如，一个包含 **3** 个上传错误消息 (**1**) 和警告消息 (**2**) 的值。一个包含 **7** 个上传错误消息 (**1**)、警告消息 (**2**) 和信息消息 (**4**) 的值。

将 Windows 事件跟踪数据发送到 CloudWatch Logs

ETW（Windows 事件跟踪）提供高效且详细的日志记录机制，供应用程序写入日志。每个 ETW 都可以启动和停止日志记录会话的会话管理器控制。每个会话都具有一个提供者以及一个或多个使用者。

1. 在 JSON 文件中，找到 ETW 部分。

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

2. 对于 LogName，请键入要上传的日志的名称。
3. 对于 Levels，指定要上传的消息类型。可以指定以下值之一：
 - **1** – 仅上传错误消息。
 - **2** – 仅上传警告消息。
 - **4** – 仅上传信息消息。

您可以将这些值组合在一起，以包含多种类型的消息。例如，一个包含 **3** 个上传错误消息 (1) 和警告消息 (2) 的值。一个包含 **7** 个上传错误消息 (1)、警告消息 (2) 和信息消息 (4) 的值。

将自定义日志（任何基于文本的日志文件）发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 CustomLogs 部分。

```
{
  "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. 对于 LogDirectoryPath，键入日志在实例上的存储路径。
3. 对于 TimestampFormat，请键入要使用的时间戳格式。有关支持的值的更多信息，请参阅 MSDN 上的 [自定义日期和时间格式字符串](#) 主题。

 Important

源日志文件必须在每个日志行开头具有时间戳，且时间戳后必须有一个空格。

4. 对于 Encoding，键入要使用的文件编码（例如 UTF-8）。有关支持的值的列表，请参阅 MSDN 上的 [Encoding 类](#) 主题。

 Note

使用编码名称，而不是显示名称。

5. (可选) 对于 Filter，键入日志名称的前缀。将此参数保留空白以监控所有文件。有关支持的值的更多信息，请参阅 MSDN 上的 [FileSystemWatcherFilter 属性](#) 主题。
6. (可选) 对于 CultureName，键入记录该时间戳的区域。如果 CultureName 为空，则它默认为您 Windows 实例当前所使用的相同区域位置。有关更多信息，请参阅 MSDN 上 [产品行为](#) 主题中表格的 Language tag 列。

 Note

不支持值 div、div-MV、hu 和 hu-HU。

7. (可选) 对于 TimeZoneKind，键入 Local 或 UTC。可以设置此参数以在日志时间戳中不包含时区信息时提供时区信息。如果此参数留空且时间戳不包括时区信息，则 CloudWatch Logs 默认为本地时区。如果时间戳已包含时区信息，则忽略此参数。
8. (可选) 对于 LineCount，在标头中键入行数以识别日志文件。例如，IIS 日志文件拥有几乎相同的标头。您可以输入 5，系统会读取日志文件标头的前三行以进行识别。在 IIS 日志文件中，第三行为日期和时间戳，但无法始终保证时间戳在日志文件之间是不同的。为此，建议包含至少一行实际日志数据以便对日志文件进行唯一指纹识别。

将 IIS 日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 IISLog 部分。

```
{
  "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. 对于 LogDirectoryPath，输入为单个站点存储的 IIS 日志所在的文件夹（例如 C:\inetpub\logs\LogFiles\W3SVCn）。

 Note

仅支持 W3C 日志格式。不支持 IIS、NCSA 和自定义格式。

3. 对于 `TimestampFormat`，请键入要使用的时间戳格式。有关支持的值的更多信息，请参阅 MSDN 上的 [自定义日期和时间格式字符串](#) 主题。
4. 对于 `Encoding`，键入要使用的文件编码（例如 UTF-8）。有关支持的值的更多信息，请参阅 MSDN 上的 [Encoding 类](#) 主题。

 Note

使用编码名称，而不是显示名称。

5. （可选）对于 `Filter`，键入日志名称的前缀。将此参数保留空白以监控所有文件。有关支持的值的更多信息，请参阅 MSDN 上的 [FileSystemWatcherFilter 属性](#) 主题。
6. （可选）对于 `CultureName`，键入记录该时间戳的区域。如果 `CultureName` 为空，则它默认为您 Windows 实例当前所使用的相同区域位置。有关支持的值的更多信息，请参阅 MSDN 上 [产品行为](#) 主题中表格的 `Language` tag 列。

 Note

不支持值 `div`、`div-MV`、`hu` 和 `hu-HU`。

7. （可选）对于 `TimeZoneKind`，输入 `Local` 或 `UTC`。可以设置此参数以在日志时间戳中不包含时区信息时提供时区信息。如果此参数留空且时间戳不包括时区信息，则 CloudWatch Logs 默认为本地时区。如果时间戳已包含时区信息，则忽略此参数。
8. （可选）对于 `LineCount`，在标头中键入行数以识别日志文件。例如，IIS 日志文件拥有几乎相同的标头。您可以输入 `5`，系统会读取日志文件标头的前五行以进行识别。在 IIS 日志文件中，第三行为日期和时间戳，但无法始终保证时间戳在日志文件之间是不同的。为此，建议包含至少一行实际日志数据以便对日志文件进行唯一指纹识别。

步骤 4：配置流程控制

每种数据类型在 `Flows` 部分中都必须具有对应的目标。例如，要将自定义日志、ETW 日志和系统日志发送到 CloudWatch Logs，请将 (`CustomLogs`, `ETW`, `SystemEventLog`), `CloudWatchLogs` 添加到 `Flows` 部分。

⚠ Warning

添加无效的步骤将阻止流。例如，如果您添加了磁盘指标步骤，但实例没有磁盘，则流中的所有步骤都将被阻止。

您可将同一个日志文件发送到多个目标。例如，要将应用程序日志发送到您在 CloudWatchLogs 部分中定义的两个不同目标，请将 ApplicationEventLog, (CloudWatchLogs, CloudWatchLogs2) 添加到 Flows 部分。

配置流程控制

1. 在 AWS.EC2.Windows.CloudWatch.json 文件中，找到 Flows 部分。

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

2. 对于 Flows，添加要上传的每种数据类型（例如 ApplicationEventLog）及其目标（例如 CloudWatchLogs）。

步骤 5：保存 JSON 内容

现在，您已完成编辑 JSON 文件。保存该文件，将文件内容粘贴到另一个窗口中的文本编辑器中。在这个过程后面的步骤中，需要使用这些文件内容。

为 Systems Manager 创建 IAM 角色

在您使用 Systems Manager Run Command 时，需要实例凭证的 IAM 角色。该角色使 Systems Manager 能够对实例执行操作。有关更多信息，请参阅 Amazon Systems Manager 用户指南中的[配置 Systems Manager 的安全角色](#)。有关如何将 IAM 角色附加到现有实例的信息，请参阅《Amazon EC2 用户指南》中的[将 IAM 角色附加到实例](#)。

验证 Systems Manager 的先决条件

在使用 Systems Manager Run Command 配置与 CloudWatch Logs 的集成之前，请确保您的实例满足最低要求。有关更多信息，请参阅 Amazon Systems Manager 用户指南中的 [Systems Manager 的先决条件](#)。

验证 Internet 访问权限

您的 Amazon EC2 Windows Server 实例和托管实例必须具有出站 Internet 访问权限，才能将日志和事件数据发送到 CloudWatch。有关如何配置 Internet 访问权限的更多信息，请参阅 Amazon VPC 用户指南中的 [Internet 网关](#)。

使用 Systems Manager Run Command 启用 CloudWatch Logs

Run Command 使您能够按需管理实例配置。您可以指定一个 Systems Manager 文档，指定一些参数，然后在一个或多个实例上执行命令。实例上的 SSM 代理负责处理命令并按指定方式配置实例。

使用 Run Command 配置与 CloudWatch Logs 的集成

1. 通过以下网址打开 Amazon EC2 控制台：<https://console.aws.amazon.com/ec2/>。
2. 访问 <https://console.aws.amazon.com/systems-manager/>，打开 SSM 控制台。
3. 在导航窗格中，选择 Run Command。
4. 选择 Run a command (运行一个命令)。
5. 对于 Command document (命令文档)，选择 AWS-ConfigureCloudWatch。
6. 对于 Target instances (目标实例)，选择要与 CloudWatch Logs 集成的实例。如果您在此列表中未看到实例，则可能未针对 Run Command 配置实例。有关更多信息，请参阅《Amazon EC2 用户指南》中的 [Systems Manager 先决条件](#)。
7. 对于 Status (状态)，选择 Enabled (已启用)。
8. 对于 Properties (属性)，请复制并粘贴您在之前任务中创建的 JSON 内容。
9. 填写剩余选填字段并选择 Run (运行)。

执行以下程序查看 Amazon EC2 控制台中的命令执行结果。

在控制台中查看命令输出

1. 选择一个命令。
2. 选择 Output (输出) 选项卡。
3. 选择 View Output (查看输出)。命令输出页面将显示命令执行的结果。

快速入门：让您的运行 Windows Server 2012 和 Windows Server 2008 的 Amazon EC2 实例将日志发送到 CloudWatch Logs

Tip

CloudWatch 包含一个可收集来自 EC2 实例和本地服务器的日志和指标的新统一代理。我们建议您使用较新的统一 CloudWatch 代理。有关更多信息，请参阅 [开始使用 CloudWatch Logs](#)。本节的其余部分将介绍较旧的 CloudWatch Logs 代理的使用。

使运行 Windows Server 2012 和 Windows Server 2008 的 Amazon EC2 实例能够将日志发送到 CloudWatch Logs

按照以下步骤启用运行 Windows Server 2012 和 Windows Server 2008 的实例，使其将日志发送到 CloudWatch Logs。

下载示例配置文件

将以下示例 JSON 文件下载到您的计算机：[AWS.EC2.Windows.CloudWatch.json](#)。您将在后续步骤中编辑该文件。

为 CloudWatch 配置 JSON 文件

您通过在 JSON 配置文件中指定选择来确定要发送到 CloudWatch 的日志。创建此文件并指定您的选择的过程可能需要 30 分钟或更长时间才能完成。完成此任务一次后，您可以在所有实例上重复使用此配置文件。

步骤

- [步骤 1：启用 CloudWatch Logs](#)
- [步骤 2：为 CloudWatch 配置设置](#)
- [步骤 3：配置要发送的数据](#)
- [步骤 4：配置流程控制](#)

步骤 1：启用 CloudWatch Logs

在 JSON 文件的顶部，将 IsEnabled 的“false”改为“true”：

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

步骤 2：为 CloudWatch 配置设置

指定凭证、区域、日志组名称和日志流命名空间。这使实例能够将日志数据发送到 CloudWatch Logs。要将同一日志数据发送到其他位置，您可以添加包含唯一 ID 的额外部分（例如，“CloudWatchLogs2”和“CloudWatchLogs3”）并为每个 ID 添加一个不同的区域。

配置设置以将日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 CloudWatchLogs 部分。

```
{
  "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. 将 AccessKey 和 SecretKey 字段留空。您将使用 IAM 角色配置凭证。
3. 对于 Region，键入要向其中发送日志数据的区域（例如 us-east-2）。
4. 对于 LogGroup，键入您的日志组的名称。此名称显示在 CloudWatch 控制台的 Log Groups（日志组）屏幕上。
5. 对于 LogStream，键入目标日志流。此名称显示在 CloudWatch 控制台的 Log Groups（日志组）> Streams（流）屏幕上。

如果使用 {instance_id}，则在默认情况下，日志流名称是该实例的实例 ID。

如果指定的日志流名称不存在，则 CloudWatch Logs 会自动创建该名称。您可以使用文字字符串、预定义变量 {instance_id}、{hostname} 和 {ip_address} 或它们的组合定义日志流名称。

步骤 3：配置要发送的数据

您可以将事件日志数据、Windows 事件跟踪 (ETW) 数据和其他日志数据发送到 CloudWatch Logs。

将 Windows 应用程序事件日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 ApplicationEventLog 部分。

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

2. 对于 Levels，指定要上传的消息类型。可以指定以下值之一：

- **1** – 仅上传错误消息。
- **2** – 仅上传警告消息。
- **4** – 仅上传信息消息。

您可以将这些值组合在一起，以包含多种类型的消息。例如，一个包含 **3** 个上传错误消息 (**1**) 和警告消息 (**2**) 的值。一个包含 **7** 个上传错误消息 (**1**)、警告消息 (**2**) 和信息消息 (**4**) 的值。

将安全日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 SecurityEventLog 部分。

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

2. 对于 Levels，键入 **7** 以上传所有消息。

将系统事件日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 SystemEventLog 部分。

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

2. 对于 Levels，指定要上传的消息类型。可以指定以下值之一：

- **1** – 仅上传错误消息。
- **2** – 仅上传警告消息。
- **4** – 仅上传信息消息。

您可以将这些值组合在一起，以包含多种类型的消息。例如，一个包含 **3** 个上传错误消息 (**1**) 和警告消息 (**2**) 的值。一个包含 **7** 个上传错误消息 (**1**)、警告消息 (**2**) 和信息消息 (**4**) 的值。

将其他类型的事件日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，添加新部分。每个部分必须有一个独立的 Id。

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

2. 对于 Id，键入要上传的日志的名称（例如 **WindowsBackup**）。
3. 对于 LogName，请键入要上传的日志的名称。您可以按如下步骤找到日志的名称。

- a. 打开事件查看器。
 - b. 在导航窗格中，选择 Applications and Services Logs (应用程序和服务日志) 。
 - c. 导航到该日志，然后选择 Actions (操作) 、 Properties (属性) 。
4. 对于 Levels，指定要上传的消息类型。可以指定以下值之一：
 - **1** – 仅上传错误消息。
 - **2** – 仅上传警告消息。
 - **4** – 仅上传信息消息。

您可以将这些值组合在一起，以包含多种类型的消息。例如，一个包含 **3** 个上传错误消息 (**1**) 和警告消息 (**2**) 的值。一个包含 **7** 个上传错误消息 (**1**)、警告消息 (**2**) 和信息消息 (**4**) 的值。

将 Windows 事件跟踪数据发送到 CloudWatch Logs

ETW (Windows 事件跟踪) 提供高效且详细的日志记录机制，供应用程序写入日志。每个 ETW 都可以启动和停止日志记录会话的会话管理器控制。每个会话都具有一个提供者以及一个或多个使用者。

1. 在 JSON 文件中，找到 ETW 部分。

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

2. 对于 LogName，请键入要上传的日志的名称。
3. 对于 Levels，指定要上传的消息类型。可以指定以下值之一：
 - **1** – 仅上传错误消息。
 - **2** – 仅上传警告消息。
 - **4** – 仅上传信息消息。

您可以将这些值组合在一起，以包含多种类型的消息。例如，一个包含 **3** 个上传错误消息 (1) 和警告消息 (2) 的值。一个包含 **7** 个上传错误消息 (1)、警告消息 (2) 和信息消息 (4) 的值。

将自定义日志（任何基于文本的日志文件）发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 CustomLogs 部分。

```
{
  "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. 对于 LogDirectoryPath，键入日志在实例上的存储路径。
3. 对于 TimestampFormat，请键入要使用的时间戳格式。有关支持的值的更多信息，请参阅 MSDN 上的 [自定义日期和时间格式字符串](#) 主题。

 Important

源日志文件必须在每个日志行开头具有时间戳，且时间戳后必须有一个空格。

4. 对于 Encoding，键入要使用的文件编码（例如 UTF-8）。有关支持的值的更多信息，请参阅 MSDN 上的 [Encoding 类](#) 主题。

 Note

使用编码名称，而不是显示名称。

5. (可选) 对于 Filter，键入日志名称的前缀。将此参数保留空白以监控所有文件。有关支持的值的更多信息，请参阅 MSDN 上的 [FileSystemWatcherFilter 属性](#) 主题。
6. (可选) 对于 CultureName，键入记录该时间戳的区域。如果 CultureName 为空，则它默认为您 Windows 实例当前所使用的相同区域位置。有关支持的值的更多信息，请参阅 MSDN 上 [产品行为](#) 主题中表格的 Language tag 列。

 Note

不支持值 div、div-MV、hu 和 hu-HU。

7. (可选) 对于 TimeZoneKind，键入 Local 或 UTC。可以设置此参数以在日志时间戳中不包含时区信息时提供时区信息。如果此参数留空且时间戳不包括时区信息，则 CloudWatch Logs 默认为本地时区。如果时间戳已包含时区信息，则忽略此参数。
8. (可选) 对于 LineCount，在标头中键入行数以识别日志文件。例如，IIS 日志文件拥有几乎相同的标头。您可以输入 5，系统会读取日志文件标头的前三行以进行识别。在 IIS 日志文件中，第三行为日期和时间戳，但无法始终保证时间戳在日志文件之间是不同的。为此，建议包含至少一行实际日志数据以便对日志文件进行唯一指纹识别。

将 IIS 日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 IISLog 部分。

```
{
  "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. 对于 LogDirectoryPath，输入为单个站点存储的 IIS 日志所在的文件夹（例如 C:\inetpub\logs\LogFiles\W3SVCn）。

 Note

仅支持 W3C 日志格式。不支持 IIS、NCSA 和自定义格式。

3. 对于 `TimestampFormat`，请键入要使用的时间戳格式。有关支持的值的更多信息，请参阅 MSDN 上的 [自定义日期和时间格式字符串](#) 主题。
4. 对于 `Encoding`，键入要使用的文件编码（例如 UTF-8）。有关支持的值的更多信息，请参阅 MSDN 上的 [Encoding 类](#) 主题。

 Note

使用编码名称，而不是显示名称。

5. （可选）对于 `Filter`，键入日志名称的前缀。将此参数保留空白以监控所有文件。有关支持的值的更多信息，请参阅 MSDN 上的 [FileSystemWatcherFilter 属性](#) 主题。
6. （可选）对于 `CultureName`，键入记录该时间戳的区域。如果 `CultureName` 为空，则它默认为您 Windows 实例当前所使用的相同区域位置。有关支持的值的更多信息，请参阅 MSDN 上 [产品行为](#) 主题中表格的 Language tag 列。

 Note

不支持值 `div`、`div-MV`、`hu` 和 `hu-HU`。

7. （可选）对于 `TimeZoneKind`，输入 `Local` 或 `UTC`。可以设置此参数以在日志时间戳中不包含时区信息时提供时区信息。如果此参数留空且时间戳不包括时区信息，则 CloudWatch Logs 默认为本地时区。如果时间戳已包含时区信息，则忽略此参数。
8. （可选）对于 `LineCount`，在标头中键入行数以识别日志文件。例如，IIS 日志文件拥有几乎相同的标头。您可以输入 `5`，系统会读取日志文件标头的前五行以进行识别。在 IIS 日志文件中，第三行为日期和时间戳，但无法始终保证时间戳在日志文件之间是不同的。为此，建议包含至少一行实际日志数据以便对日志文件进行唯一指纹识别。

步骤 4：配置流程控制

每种数据类型在 Flows 部分中都必须具有对应的目标。例如，要将自定义日志、ETW 日志和系统日志发送到 CloudWatch Logs，请将 (`CustomLogs`, `ETW`, `SystemEventLog`), `CloudWatchLogs` 添加到 Flows 部分。

 Warning

添加无效的步骤将阻止流。例如，如果您添加了磁盘指标步骤，但实例没有磁盘，则流中的所有步骤都将被阻止。

您可将同一个日志文件发送到多个目标。例如，要将应用程序日志发送到您在 CloudWatchLogs 部分中定义的两个不同目标，请将 ApplicationEventLog, (CloudWatchLogs, CloudWatchLogs2) 添加到 Flows 部分。

配置流程控制

1. 在 AWS.EC2.Windows.CloudWatch.json 文件中，找到 Flows 部分。

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

2. 对于 Flows，添加要上传的每种数据类型（例如 ApplicationEventLog）及其目标（例如 CloudWatchLogs）。

现在，您已完成编辑 JSON 文件。您将在后面的步骤中用到它。

启动该代理

要让运行 Windows Server 2012 或 Windows Server 2008 的 Amazon EC2 实例将日志发送到 CloudWatch Logs，请使用 EC2Config 服务 (EC2Config.exe)。您的实例应具有 EC2Config 4.0 或更高版本并且您可以使用此过程。

使用 EC2Config 4.x 配置 CloudWatch

1. 检查您在此过程前面编辑的 AWS.EC2.Windows.CloudWatch.json 文件的编码。只支持不含 BOM 的 UTF-8 编码。然后将文件保存在 Windows Server 2008 - 2012 R2 实例上的以下文件夹内：C:\Program Files\Amazon\SSM\Plugins\awsCloudWatch\。

2. 使用 Windows 服务控制面板或者使用以下 PowerShell 命令，来启动或重新启动 SSM Agent (AmazonSSMAgent.exe)：

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

在重新启动 SSM Agent 后，代理会检测配置文件并且为 CloudWatch 集成配置实例。如果您更改本地配置文件中的参数和设置，则必须重新启动 SSM Agent 来使更改生效。如果希望在实例上禁用 CloudWatch 集成，请将 IsEnabled 更改为 false 并保存在配置文件中所做的更改。

报告 CloudWatch Logs 代理的状态

使用以下步骤报告 EC2 实例上 CloudWatch Logs 代理的状态。

报告代理状态

1. 连接到 EC2 实例。有关详细信息，请参阅《Amazon EC2 用户指南》中的[连接到您的实例](#)。

有关连接问题的更多信息，请参阅《Amazon EC2 用户指南》中的[排查实例的连接问题](#)

2. 在命令提示窗口中，键入以下命令：

```
sudo service awslogs status
```

如果您运行的是 Amazon Linux 2，请键入以下命令：

```
sudo service awslogsd status
```

3. 使用 CloudWatch Logs 代理检查 /var/log/awslogs.log 文件中是否有任何错误、警告或问题。

启动 CloudWatch Logs 代理

如果您的 EC2 实例上的 CloudWatch Logs 代理在安装后没有自动启动，或者您停止了该代理，则可以使用以下步骤启动代理。

启动代理

1. 连接到 EC2 实例。有关详细信息，请参阅《Amazon EC2 用户指南》中的[连接到您的实例](#)。

有关连接问题的更多信息，请参阅《Amazon EC2 用户指南》中的[排查实例的连接问题](#)。

2. 在命令提示窗口中，键入以下命令：

```
sudo service awslogs start
```

如果您运行的是 Amazon Linux 2，请键入以下命令：

```
sudo service awslogsd start
```

停止 CloudWatch Logs 代理

使用以下步骤停止 EC2 实例上的 CloudWatch Logs 代理。

停止代理

1. 连接到 EC2 实例。有关详细信息，请参阅《Amazon EC2 用户指南》中的[连接到您的实例](#)。

有关连接问题的更多信息，请参阅《Amazon EC2 用户指南》中的[排查实例的连接问题](#)。

2. 在命令提示窗口中，键入以下命令：

```
sudo service awslogs stop
```

如果您运行的是 Amazon Linux 2，请键入以下命令：

```
sudo service awslogsd stop
```

CloudWatch Logs 代理参考

Important

本节为使用已弃用的 CloudWatch Logs 代理的用户提供参考。如果使用 Instance Metadata Service Version 2 (IMDSv2)，则必须使用新的统一 CloudWatch 代理。但是，即使您不使用 IMDSv2，我们也强烈建议您使用较新的统一 CloudWatch 代理，而不是已弃用的 CloudWatch Logs 代理。有关较新的统一 CloudWatch 代理的信息，请参阅[使用 CloudWatch 代理从 Amazon EC2 实例和本地服务器中收集指标和日志](#)。有关从已弃用的 CloudWatch Logs 代理迁移到统一代理的信息，请参阅[使用向导创建 CloudWatch 代理配置文件](#)。

CloudWatch Logs 代理提供一种从 Amazon EC2 实例将日志数据发送到 CloudWatch Logs 的自动化方式。该代理包括以下组件：

- 一个将日志数据推送到 CloudWatch Logs 的 Amazon CLI 插件。
- 一个启动进程以将数据推送到 CloudWatch Logs 的脚本（守护进程）。
- 一个确保该守护程序始终运行的 cron 作业。

代理配置文件

CloudWatch Logs 代理配置文件描述 CloudWatch Logs 代理所需的信息。代理配置文件的 [general] 一节定义适用于所有日志流的通用配置。[logstream] 一节定义将本地文件发送到远程日志流所必需的信息。您可以有多个 [logstream] 节，但是每一节的名称在该配置文件中都必须唯一，如 [logstream1]、[logstream2] 等等。[logstream] 值和日志文件第一行数据共同定义日志文件的标识。

```
[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

指定状态文件的存储位置。

logging_config_file

(可选) 指定代理日志记录配置文件的位置。如果您未在此处指定代理日志记录配置文件，系统将使用默认文件 `awslogs.conf`。如果使用脚本安装代理，默认文件位置是 `/var/awslogs/etc/awslogs.conf`，如果使用 rpm 安装代理，默认文件位置是 `/etc/awslogs/awslogs.conf`。此文件采用 Python 配置文件格式 (<https://docs.python.org/2/library/logging.config.html#logging-config-fileformat>)。可自定义具有以下名称的日志记录程序。

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

尽管默认值为 INFO，以下示例仍会将阅读者和发布者的级别更改为 WARNING。

```
[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

在设置为 true (默认值) 时，启用 gzip http 内容编码来将压缩的负载发送到 CloudWatch Logs。这可以降低 CPU 使用率，减少 NetworkOut，并降低放入延迟。要禁用该功能，请在 CloudWatch Logs 代理配置文件的 [general] 节中添加 use_gzip_http_content_encoding = false，然后重新启动代理。

Note

此设置只在 awscli-cwlogs 版本 1.3.3 和更高版本中可用。

log_group_name

指定目标日志组。如果还没有日志组，则会自动创建一个日志组。日志组名称的长度可介于 1 和 512 个字符之间。允许的字符包括 a-z、A-Z、0-9、“_” (下划线)、“-” (连字符)、“/” (正斜杠) 和“.” (句点)。

log_stream_name

指定目标日志流。您可以使用文字字符串或预定义的变量 ({instance_id}、{hostname}、{ip_address})，或这两者的组合来定义日志流名称。如果还没有日志流，则会自动创建一个日志流。

datetime_format

指定如何从日志提取时间戳。时间戳用于检索日志事件和生成指标。如果未提供 `datetime_format`，则将当前时间用于每个日志事件。如果提供的 `datetime_format` 值对于给定日志消息无效，则使用时间戳成功解析的最近日志事件的时间戳。如果不存在以前的日志事件，则使用当前时间。

下面列出了常见 `datetime_format` 代码。您也可以使用 Python `datetime.strptime()` 支持的所有 `datetime_format` 代码。时区偏移量 (`%z`) 也受支持 (即使 Python 3.2 之前的版本都不支持它)，`[+-]HHMM`，不带冒号 (`:`)。有关更多信息，请参阅 [strptime\(\) 和 strftime\(\) 行为](#)。

`%y`：年份，以零填充的十进制数字表示，不包括代表世纪的数字。00, 01, ..., 99

`%Y`：年份，以十进制数字形式表示且包括表示世纪的数字。如 1970、1988、2001、2013

`%b`：月份，使用区域设置的缩写名称形式。Jan、Feb...Dec (`en_US`)；

`%B`：月份，使用区域设置的完整名称形式。January, February...December (`en_US`)；

`%m`：月份，使用以零填充的十进制数字形式。01, 02, ..., 12

`%d`：月份中的日期，使用以零填充的十进制数字形式。01, 02, ..., 31

`%H`：小时 (24 小时制)，使用以零填充的十进制数字形式。00, 01, ..., 23

`%I`：小时 (12 小时制)，使用以零填充的十进制数字形式。01, 02, ..., 12

`%p`：区域设置中等效于 AM 或 PM 的表示形式。

`%M`：分钟，使用以零填充的十进制数字形式。00, 01, ..., 59

`%S`：秒，使用以零填充的十进制数字形式。00, 01, ..., 59

`%f`：微秒，在左边使用以零填充的十进制数字形式。000000, ..., 999999

`%z`：UTC 偏移量，采用 `+HHMM` 或 `-HHMM` 形式。`+0000`、`-0400`、`+1030`

示例格式：

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

指定日志事件时间戳的时区。支持的两个值为 UTC 和 LOCAL。默认值为 LOCAL，如果无法基于 `datetime_format` 推断时区，则将使用此默认值。

文件

指定您要推送到 CloudWatch Logs 的日志文件。File 可以指向一个特定文件或多个文件（使用通配符，如 `/var/log/system.log*`）。根据文件修改时间，只有最新文件才会推送到 CloudWatch Logs。我们建议您使用通配符指定同一类型的一系列文件（如 `access_log.2014-06-01-01`、`access_log.2014-06-01-02` 等）而不是多个类型的文件（如 `access_log_80` 和 `access_log_443`）。要指定多个类型的文件，请在配置文件中再添加一个日志流条目，让每一种日志文件转到不同的日志流。不支持压缩文件。

file_fingerprint_lines

指定用于识别文件的行范围。有效值是一个数字或两个用短划线分隔的数字，如“1”、“2-5”。默认值是“1”，因此第一行用于计算指纹。除非所有指定行都可用，否则指纹不会发送到 CloudWatch Logs。

multi_line_start_pattern

指定用于识别日志消息开头的模式。日志消息由与模式匹配的行以及与模式不匹配的任何以下行组成。有效值为正则表达式或 `{datetime_format}`。使用 `{datetime_format}` 时，应指定 `datetime_format` 选项。默认值为“`^[^\s]`”，因此以非空格字符开头的任何行都会使上一个日志消息结束，并开始新的日志消息。

initial_position

指定从何处开始读取数据（`start_of_file` 或 `end_of_file`）。默认位置是 `start_of_file`。仅当日志流没有持续状态时才会使用它。

编码

指定日志文件的编码，以便正确读取该文件。默认编码为 `utf_8`。Python `codecs.decode()` 支持的编码可在此处使用。

Warning

指定错误的编码可能导致数据丢失，因为无法解码的字符将被其他字符替代。

以下是一些常见编码：

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

指定日志事件批量处理的时间段。最小值为 5000ms，默认值为 5000ms。

batch_count

指定批处理中的日志事件的最大数量（最大为 10000）。默认值是 10000。

batch_size

指定批处理中的日志事件的最大大小（以字节为单位，最大为 1048576 字节）。默认值为 1048576 字节。此大小的计算方式是 UTF-8 格式的所有事件消息之和加上代表每个日志事件的 26 字节。

结合使用 CloudWatch Logs 代理与 HTTP 代理

您可以结合使用 CloudWatch Logs 代理与 HTTP 代理。

Note

在 awslogs-agent-setup.py 版本 1.3.8 或更高版本中支持 HTTP 代理。

结合使用 CloudWatch Logs 代理与 HTTP 代理

1. 请执行以下操作之一：

- a. 对于新安装的 CloudWatch Logs 代理，请运行以下命令：

```
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
```

为了维护在 EC2 实例上对 Amazon EC2 元数据服务的访问，请使用 `--no-proxy 169.254.169.254`（建议）。有关更多信息，请参阅《Amazon EC2 用户指南》中的[实例元数据和用户数据](#)。

在 `http-proxy` 和 `https-proxy` 的值中，您指定完整 URL。

- b. 对于 CloudWatch Logs 代理的现有安装，请编辑 `/var/awslogs/etc/proxy.conf`，然后添加您的代理：

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

2. 重新启动代理以使更改生效：

```
sudo service awslogs restart
```

如果您使用的是 Amazon Linux 2，请使用以下命令重新启动代理：

```
sudo service awslogsd restart
```

划分 CloudWatch Logs 代理配置文件

如果您正在将 `awslogs-agent-setup.py` 版本 1.3.8 或更高版本与 `awscli-cwlogs` 1.3.3 或更高版本结合使用，可以通过在 `/var/awslogs/etc/config/` 目录中创建额外配置文件，独立于其他组件来导入各种组件的不同流配置。在 CloudWatch Logs 代理启动时，将在这些额外的配置文件中包含所有流配置。`[general]` 节中的配置属性必须在主配置文件 (`/var/awslogs/etc/awslogs.conf`) 中定义，并且在 `/var/awslogs/etc/config/` 中找到的任何额外配置文件中将被忽略。

如果您是使用 rpm 安装的代理，因此没有 `/var/awslogs/etc/config/` 目录，则可以使用 `/etc/awslogs/config/` 目录代替。

重新启动代理以使更改生效：

```
sudo service awslogs restart
```

如果您使用的是 Amazon Linux 2，请使用以下命令重新启动代理：

```
sudo service awslogsd restart
```

CloudWatch Logs 代理常见问题

支持哪些类型的文件轮换？

支持以下文件轮换机制：

- 用数字后缀为现有日志文件命名，然后重新创建原始的空日志文件。例如，`/var/log/syslog.log` 重命名为 `/var/log/syslog.log.1`。如果 `/var/log/syslog.log.1` 从上次轮换起就已存在，则重命名为 `/var/log/syslog.log.2`。
- 在创建副本后截断原始日志文件。例如，`/var/log/syslog.log` 复制到 `/var/log/syslog.log.1`，会截断 `/var/log/syslog.log`。这种情况下可能会有数据丢失，因此请谨慎使用这种文件轮换机制。
- 使用与旧文件相同的通用模式创建新文件。例如，`/var/log/syslog.log.2014-01-01` 仍然保留，将创建 `/var/log/syslog.log.2014-01-02`。

文件的指纹（源 ID）是通过将日志流键和文件内容第一行进行哈希处理计算得到的。要覆盖此行为，可以使用 `file_fingerprint_lines` 选项。当文件进行轮换时，新文件应该有新内容，旧文件不应追加内容；代理将在完成旧文件的读取后推送新文件。

如何确定我使用的是哪个版本的代理？

如果您使用设置脚本安装了 CloudWatch Logs 代理，则可以使用 `/var/awslogs/bin/awslogs-version.sh` 来检查您使用的是哪个版本的代理。它会打印代理的版本及其主要依赖关系。如果您用 yum 安装了 CloudWatch Logs 代理，则可以使用“`yum info awslogs`”和“`yum info aws-cli-plugin-cloudwatch-logs`”来检查 CloudWatch Logs 代理和插件的版本。

日志条目如何转换为日志事件？

日志事件包含两个属性：事件发生时的时间戳，以及原始日志消息。默认情况下，以非空格字符开头的任何行都会使上一个日志消息结束（如果存在），并开始新的日志消息。要覆

盖此行为，可以使用 `multi_line_start_pattern`，与模式匹配的任何行都开始新的日志消息。模式可以是任何正则表达式或“`{datetime_format}`”。例如，如果每个日志消息的第一行都包含类似于“2014-01-02T13:13:01Z”的时间戳，则 `multi_line_start_pattern` 可以设置为“`\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}Z`”。要简化配置，可以在指定 `datetime_format` 选项的情况下使用“`{datetime_format}`”变量。对于同一个示例，如果 `datetime_format` 设置为“`%Y-%m-%dT%H:%M:%S%z`”，则 `multi_line_start_pattern` 可以仅仅是“`{datetime_format}`”。

如果未提供 `datetime_format`，则将当前时间用于每个日志事件。如果提供的 `datetime_format` 对于给定日志消息无效，则使用时间戳成功解析的最近日志事件的时间戳。如果不存在以前的日志事件，则使用当前时间。当日志事件回退到当前时间或上一个日志事件的时间时，会记录一个警告消息。

时间戳用于检索日志事件和生成指标，因此，如果您指定错误的格式，则可能无法检索日志事件，生成错误的指标。

日志事件如何批处理？

满足以下任意条件时，表示批次已满并且将发布：

1. 从添加第一个日志事件以来，时间已经过了 `buffer_duration`。
2. 累积的日志事件小于 `batch_size`，但添加新的日志事件则会超出 `batch_size`。
3. 日志事件的数量已达到 `batch_count`。
4. 批处理中的日志事件的时间跨度不超过 24 小时，但添加新日志事件会超出 24 小时的限制。

什么原因可能导致跳过或截断日志条目、日志事件或批次？

为遵循 `PutLogEvents` 操作的限制，需注意，以下问题可能导致跳过日志事件或批次。

Note

当跳过数据时，CloudWatch Logs 代理会在其日志中写入一条警告。

1. 如果日志事件的大小超过 256 KB，则将完全跳过日志事件。
2. 如果日志事件的时间戳晚于未来 2 小时，则跳过日志事件。
3. 如果日志事件的时间戳早于过去 14 天，则跳过日志事件。
4. 如果任何日志事件的存在时间超过日志组的保留期，则跳过整个批次。
5. 如果单个 `PutLogEvents` 请求中的一批日志事件时间跨度超过 24 小时，则 `PutLogEvents` 操作将失败。

停止代理会导致数据丢失/重复条目吗？

只要状态文件可用，且从上次运行以来没有发生文件轮换，则不会。CloudWatch Logs 代理可以从它停止的地方启动，并继续推送日志数据。

我可以将来自相同或不同主机的不同日志文件指向同一个日志流吗？

不支持配置多个日志源将数据发送到单个日志流。

代理调用哪些 API（或我应该将哪些操作添加到 IAM 策略中）？

CloudWatch Logs 代理需要权限才能执行

CreateLogGroup、CreateLogStream、DescribeLogStreams、DescribeLogGroup、PutLogEvents 和 PutRetentionPolicy 操作。如果您使用最新的代理，则不需要 DescribeLogStreams。请参阅以下 IAM 策略示例。

JSON

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

我不希望 CloudWatch Logs 代理自动创建日志组或日志流。我如何阻止代理重新创建日志组和日志流？

在您的 IAM 策略中，您可以限制该代理仅执行以下操作：DescribeLogStreams 和 PutLogEvents。

在撤销代理的 `CreateLogGroup` 和 `CreateLogStream` 权限之前，请确保创建希望代理使用的日志组和日志流。如果日志代理没有 `CreateLogGroup` 和 `CreateLogStream` 权限，它将无法在您创建的日志组中创建日志流。

在进行故障排除时我应当查看哪些日志？

代理安装日志位于 `/var/log/awslogs-agent-setup.log`，代理日志位于 `/var/log/awslogs.log`。

快速入门：使用 Amazon CloudFormation 来开始使用 CloudWatch Logs

Amazon CloudFormation 支持以 JSON 格式描述和预置 Amazon 资源。此方法的优点包括能够将 Amazon 资源集合作为一个单元进行管理，并且可以轻松地跨区域复制 Amazon 资源。

使用 Amazon CloudFormation 预置 Amazon 时，可以创建描述要使用的 Amazon 资源的模板。以下示例是一个模板代码段，它创建一个日志组和一个指标筛选器，可以统计 404 个事件，并将此计数发送到日志组。

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

```
}  
}
```

这是一个基本示例。您可以使用 Amazon CloudFormation 设置更丰富的 CloudWatch Logs 部署。有关模板示例的更多信息，请参阅 Amazon CloudFormation 用户指南中的 [Amazon CloudWatch Logs 模板代码段](#)。有关开始使用的更多信息，请参阅 Amazon CloudFormation 用户指南中的 [开始使用 Amazon CloudFormation](#)。

将 CloudWatch Logs 与 Amazon SDK 结合使用

Amazon 软件开发工具包 (SDK) 适用于许多常用编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地了解其首选语言构建应用程序。

SDK 文档

[Amazon CLI](#)

[适用于 Java 的 Amazon SDK](#)

[适用于 JavaScript 的 Amazon SDK](#)

[适用于 .NET 的 Amazon SDK](#)

[适用于 PHP 的 Amazon SDK](#)

[Amazon Tools for PowerShell](#)

[适用于 Python \(Boto3\) 的 Amazon SDK](#)

[适用于 Ruby 的 Amazon SDK](#)

[适用于 SAP ABAP 的 Amazon SDK](#)

有关 CloudWatch Logs 的特定示例，请参阅 [使用 Amazon SDK 的 CloudWatch Logs 代码示例](#)。

使用 CloudWatch Logs Insights 分析日志数据

借助 CloudWatch Logs Insights，您可以以交互方式搜索和分析 Amazon CloudWatch Logs 中的日志数据。您可以执行查询以帮助您更高效地响应操作问题。如果出现问题，您可以使用 CloudWatch Logs Insights 确定潜在原因并验证已部署的修复程序。

CloudWatch Logs Insights 支持三种可用于查询的查询语言：

- 一种专门构建的 Logs Insights 查询语言 (Logs Insights QL)，包含一些简单但功能强大的命令。
- OpenSearch Service 管道处理语言 (PPL)。OpenSearch PPL 使您能够使用由管道 (|) 分隔的一组命令来分析日志。

借助 OpenSearch PPL，您可以使用通过管道连接在一起的命令来检索、查询和分析数据，从而更轻松地理解和编写复杂的查询。该语法使命令链能够转换和处理数据。使用 PPL，您可以对数据进行筛选和聚合，并使用一组丰富的数学、字符串、日期、条件和其他函数进行分析。

- OpenSearch Service 结构化查询语言 (SQL)。借助 OpenSearch SQL 查询，您可以以声明方式分析日志。您可以使用 SELECT、FROM、WHERE、GROUP BY、HAVING 等命令，以及 SQL 中提供的各种其他命令和函数。您可以跨日志组执行 JOIN 操作，使用子查询关联日志中的数据，并使用丰富的 JSON、数学、字符串、条件和其他 SQL 函数集对日志执行强大的分析。

使用 SQL 或 PPL 命令时，请务必用反引号将包含特殊字符（非字母和非数字）的字段括起来，才能成功查询。例如，对 @message、Operation.Export 和 Test::Field 使用反引号。纯字母名称的字段无需使用反引号。

CloudWatch Logs Insights 提供以下功能，这些功能可与任何查询语言一起使用。

- 自动发现 Amazon 服务（例如 Amazon Route 53、Amazon Lambda、Amazon CloudTrail 和 Amazon VPC）的日志中的 [日志字段](#)，以及以 JSON 格式发出日志事件的任何应用程序或自定义日志中的字段。
- 创建 [字段索引](#) 可以降低成本并加快查询速度，尤其适用于处理大量日志组或日志事件的查询。在为日志事件中常见的字段创建字段索引后，即可在查询中使用这些索引。查询会跳过处理已知不包含索引字段的日志事件，并处理较少的数据。

Note

filterIndex 命令仅在 Logs Insights QL 中可用。

- [检测和分析日志事件中的模式](#)。模式是在日志字段中重复出现的共有的文本结构。查看查询结果时，您可以选择模式选项卡来查看 CloudWatch Logs 根据结果示例找到的模式。
- [保存查询](#)、查看查询历史记录并重新运行已保存的查询。这可以帮助您在需要时运行复杂的查询，而无需每次要运行它们时都重新创建它们。
- [向控制面板添加查询](#)。
- [使用 Amazon Key Management Service 加密查询结果](#)。
- [使用自然语言生成查询](#)可让您使用自然语言创建 CloudWatch Logs Insights 查询。您可以询问有关您要查找的数据的问题或描述该数据，然后 AI 会根据您的提示生成查询，并逐行说明查询的工作原理。

以下 CloudWatch Logs Insights 功能仅在使用 Logs Insights QL 时受支持：

- 查询[不频繁访问日志类](#)中的日志。
- [比较查询](#)，将日志组中的日志事件与先前时间段的日志事件进行比较。
- [filterIndex 命令](#)，强制查询仅尝试扫描包含您指定的字段索引的日志事件。

 Important

CloudWatch Logs Insights 无法访问时间戳早于日志组创建时间的日志事件。

如果您登录了在 CloudWatch 跨账户可观察性中设置为监控账户的账户，则可以对与该监控账户关联的源账户中的日志组运行 CloudWatch Logs Insights 查询。您可以运行一个查询，查询位于不同账户中的多个日志组。有关更多信息，请参阅[CloudWatch 跨账户可观察性](#)。

使用 Logs Insights QL 创建查询时，您还可以使用自然语言创建 CloudWatch Logs Insights 查询。要执行此操作，请询问或描述您要查找的数据。此 AI 辅助功能可按照您的提示生成查询，并逐行说明查询的工作原理。有关更多信息，请参阅[使用自然语言生成和更新 CloudWatch Logs Insights 查询](#)。

如果使用任何支持的查询语言的查询尚未完成，则会在 60 分钟后超时。查询结果的有效期为七天。

CloudWatch Logs Insights 查询将根据所查询的数据量收费，而不管查询语言如何。有关更多信息，请参阅[Amazon CloudWatch 定价](#)。

您可以使用 CloudWatch Logs Insights 搜索在 2018 年 11 月 5 日或之后发送到 CloudWatch Logs 的日志数据。

Important

如果您的网络安全团队不允许使用 Web 套接字，则您目前无法访问 CloudWatch 控制台的 CloudWatch Logs Insights 部分。您可以通过 API 使用 CloudWatch Logs Insights 查询功能。有关更多信息，请参阅 Amazon CloudWatch Logs API Reference 中的 [StartQuery](#)。

目录

- [支持的查询语言](#)
- [使用自然语言生成和更新 CloudWatch Logs Insights 查询](#)
- [支持的日志和发现的字段](#)
- [创建字段索引以提高查询性能并减少扫描量](#)
- [模式分析](#)
- [保存并重新运行 CloudWatch Logs Insights 查询](#)
- [将查询添加到控制面板或导出查询结果](#)
- [查看正在运行的查询或查询历史记录](#)
- [使用 Amazon Key Management Service 加密查询结果](#)
- [根据 CloudWatch Logs Insights 查询结果生成自然语言摘要](#)

支持的查询语言

以下部分列出了每种查询语言支持的命令。它们还介绍语法格式并提供示例查询。

主题

- [CloudWatch Logs Insights 查询语言 \(Logs Insights QL\)](#)
- [OpenSearch 管道处理语言 \(PPL\)](#)
- [OpenSearch 结构化查询语言 \(SQL\)](#)

CloudWatch Logs Insights 查询语言 (Logs Insights QL)

本节包括 Logs Insights QL 命令和函数的完整文档。它还包括此语言的示例查询。

有关您可以使用的其他查询语言的信息，请参阅 [OpenSearch Service PPL](#)、[OpenSearch Service SQL](#) 和 [CloudWatch Metrics Insights](#)。

主题

- [CloudWatch Logs Insights 语法查询语法](#)
- [开始使用 Logs Insights QL：查询教程](#)
- [示例查询](#)
- [与之前的时间范围进行比较（差异）](#)
- [在图形中可视化日志数据](#)

CloudWatch Logs Insights 语法查询语法

本节详细介绍 Logs Insights QL。查询语法支持不同的函数和运算，包括但不限于常规函数、算术和比较运算以及正则表达式。

Important

为了避免因运行大型查询而产生过高的费用，请牢记以下最佳实践：

- 仅为每个查询选择必要的日志组。
- 始终为查询指定尽可能窄的时间范围。
- 当使用控制台运行查询时，请在关闭 CloudWatch Logs Insights 控制台页面之前取消所有查询。否则，查询将继续运行直至完成。
- 当您将 CloudWatch Logs Insights 小组件添加到控制面板时，请确保仪表板不会以高频率刷新，因为每次刷新都会启动一个新查询。

要创建包含多个命令的查询，请使用竖线字符 (|) 分隔命令。

要创建包含注释的查询，请使用哈希字符 (#) 对注释进行分隔。

Note

CloudWatch Logs Insights 会自动发现多种不同日志类型的字段，并生成以 @ 字符开头的字段。有关这些字段的更多信息，请参阅 Amazon CloudWatch 用户指南中的[支持的日志和搜索到的字段](#)。

下表简要介绍了每个命令。下表对每条命令进行了更全面的描述，并附有示例。

Note

标准日志类中的日志组支持所有 Logs Insights QL 查询命令。不频繁访问日志类中的日志组支持除 `pattern`、`diff` 和 `unmask` 之外的所有 Logs Insights QL 查询命令。

<u>anomaly</u>	使用机器学习识别日志数据中的异常模式。
<u>display</u>	在查询结果中显示一个或多个特定字段。
<u>fields</u>	在查询结果中显示多个特定字段，并支持函数和操作，可用于修改字段值和创建用于查询的新字段。
<u>filter</u>	筛选查询，以仅返回与一个或多个条件匹配的日志事件。
<u>filterIndex</u>	强制查询尝试仅扫描已在字段索引提及的字段上编制索引并且还包含该字段索引的值的日志组。这样可以减少扫描量，因为它只尝试扫描这些日志组中包含此字段索引的查询中指定的值的日志事件。 不频繁访问日志类中的日志组不支持此命令。
<u>pattern</u>	自动将您的日志数据划分为不同模式。模式是在日志字段中重复出现的共有的文本结构。CloudWatch Logs Insights 为您提供了分析在日志事件中发现的模式的方法。有关更多信息，请参阅 模式分析 。
<u>diff</u>	将在您请求的时间段内找到的日志事件与之前相同长度的时间段内的日志事件进行比较，以便您可以查找趋势并确定某些日志事件是否是新的。
<u>parse</u>	从日志字段中提取数据，以创建可以在查询中处理的提取字段。 parse 同时支持使用通配符和正则表达式的 glob 模式。
<u>sort</u>	按升序 (asc) 或降序 (desc) 顺序显示返回的日志事件。
<u>SOURCE</u>	在查询中包含 SOURCE 参数是一种根据日志组名称前缀、账户标识符和日志组类指定要包含在查询中的大量日志组的有用方法。只有当您在 Amazon CLI 中或以编程方式创建查询时才支持此命令，在 CloudWatch 控制台中不支持。
<u>stats</u>	使用日志字段值计算聚合统计数据。

<u>limit</u>	指定您希望查询返回的最大日志事件数。对于 sort 返回“前 20 个”或“最近 20 个”结果很有用。
<u>dedup</u>	根据您指定的字段中的特定值删除重复的结果。
<u>unmask</u>	显示由于数据保护策略而屏蔽部分内容的日志事件的所有内容。有关日志组中数据保护的更多信息，请参阅 通过屏蔽帮助保护敏感的日志数据 。
<u>unnest</u>	对输入列表进行扁平化以生成多条记录，列表中的每个元素都有一条记录。
<u>其他操作和函数</u>	CloudWatch Logs Insights 还支持许多比较、算术、日期时间、数值、字符串、IP 地址以及常规函数和操作。

以下各部分提供有关 CloudWatch Logs Insights 查询命令的更多详细信息。

主题

- [日志类中支持的 Logs Insights QL 命令](#)
- [异常](#)
- [display](#)
- [fields](#)
- [筛选](#)
- [filterIndex](#)
- [SOURCE](#)
- [模式](#)
- [diff](#)
- [parse](#)
- [排序](#)
- [stats](#)
- [limit](#)
- [dedup](#)
- [unmask](#)
- [unnest](#)

- [布尔值、比较、数值、日期时间和其他函数](#)
- [包含特殊字符的字段](#)
- [在查询中使用别名和注释](#)

日志类中支持的 Logs Insights QL 命令

标准日志类中的日志组支持所有 Logs Insights QL 查询命令。不频繁访问日志类中的日志组支持除 `pattern`、`diff`、`filterIndex` 和 `unmask` 之外的所有查询命令。

异常

使用 `anomaly` 可以通过机器学习自动识别日志数据中的异常模式和潜在问题。

`anomaly` 命令扩展了现有 `pattern` 功能，并利用高级分析来帮助识别日志数据中的潜在异常。您可以使用 `anomaly` 自动在日志中发现异常模式或行为，从而减少识别和解决操作问题所需的时间。

`anomaly` 命令与 [pattern](#) 命令配合使用，首先识别日志模式，然后检测这些模式中的异常。您也可以将 `anomaly` 与 [filter](#) 或 [sort](#) 命令结合使用，将异常检测集中在数据的特定子集上。

`anomaly` 命令输入

`anomaly` 命令通常在 [pattern](#) 命令之后使用，用于分析日志数据中发现的模式。该命令不需要额外的参数，它会分析查询中前面命令的输出。

已识别的异常类型

`anomaly` 命令可识别五种不同类型的异常：

- 模式频率异常：特定日志模式的异常频率，例如当应用程序开始生成比平时更多的错误消息时。
- 新模式异常：以前未曾见过的日志模式，可能表明日志中出现了新的错误或消息类型。
- 令牌变异异常：日志消息内容的意外更改，可能表示预期日志格式存在异常变化。
- 数字令牌异常：日志中数值的异常变化，可以帮助检测潜在性能问题或意外的指标变化。
- HTTP 错误代码异常：与 HTTP 错误响应相关的模式，在监控 Web 应用程序和 API 时特别有用。

`anomaly` 命令输入

`anomaly` 命令会保留输入数据中的所有字段并添加异常检测结果，以帮助识别日志数据中的异常模式。

示例

以下命令可识别日志数据中的模式，然后检测这些模式中的异常：

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

`anomaly` 命令可以与筛选结合使用，以专注于特定的日志类型：

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

`anomaly` 命令可以与排序结合使用以整理结果：

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

display

使用 `display` 在查询结果中显示一个或多个特定字段。

`display` 命令仅显示您指定的字段。如果您的查询包含多个 `display` 命令，则查询结果仅会显示您在最终 `display` 命令中指定的一个或多个字段。

示例：显示一个字段

该代码片段显示了一个查询示例，其使用解析命令从 `@message` 提取数据以创建提取字段 `loggingType` 和 `loggingMessage`。该查询将返回 `loggingType` 的值为 `ERROR`（错误）的所有日志事件。`display` 在查询结果中仅显示 `loggingMessage` 的值。

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


 Tip

在查询中只使用 `display` 一次。如果您在查询中多次使用 `display`，则查询结果会显示在最后使用的 `display` 命令中指定的字段。

fields

使用 `fields` 在查询结果中显示特定字段。

如果您的查询包含多个 `fields` 命令，且不包括 `display` 命令，则会显示在 `fields` 命令中指定的所有字段。

示例：显示特定字段

以下示例显示一个查询，将返回 20 个日志事件并按降序显示这些事件。查询结果将显示 `@timestamp` 和 `@message` 的值。

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

当您希望使用 `fields` 支持的不同函数和操作来修改字段值并创建可在查询中使用的新字段时，请使用 `fields` 而不是 `display`。

您可以将 `fields` 命令与关键字 `as` 配合使用，以创建使用您的日志事件中的字段和函数的提取字段。例如，`fields ispresent as isRes` 将创建一个名为 `isRes` 的提取字段，并且该提取字段可在查询的其余部分中使用。

筛选

使用 `filter` 获取与一个或多个条件匹配的日志事件。

示例：使用一个条件筛选日志事件

该代码片段显示了一个查询示例，其将返回 `range` 的值大于 3000 的所有日志事件。该查询将结果限制为 20 个日志事件，并按 `@timestamp` 和降序对日志事件进行排序。

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

```
| limit 20
```

示例：使用多个条件筛选日志事件

您可以使用关键字 `and` 和 `or` 组合多个条件。

该代码片段显示了一个查询示例，其将返回 `range` 的值大于 3000 以及 `accountId` 的值等于 123456789012 的日志事件。该查询将结果限制为 20 个日志事件，并按 `@timestamp` 和降序对日志事件进行排序。

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

索引字段和 filter 命令

如果已为日志组创建字段索引，则可以利用这些字段索引来提高 `filter` 查询的效率并减少扫描量。例如，假设您为 `requestId` 创建了一个字段索引。然后，如果任何针对该日志组的 CloudWatch Logs Insights 查询包含 `filter requestId = value` 或 `filter requestId IN [value, value, ...]`，则将尝试跳过处理已知不包含索引字段的日志事件。通过尝试仅扫描已知包含该索引字段的日志事件，可以减少扫描量，查询速度更快。

有关字段索引和如何创建它们的更多信息，请参阅[创建字段索引以提高查询性能并减少扫描量](#)。

Important

只有包含 `filter fieldName = ...` 和 `filter fieldName IN ...` 的查询将受益于字段索引改进。包含 `filter fieldName like` 的查询不使用索引，并且始终扫描所选日志组中的所有日志事件。

示例：使用索引查找与特定请求 ID 相关的日志事件

此示例假设您已在 `requestId` 上创建了字段索引。对于使用此字段索引的日志组，查询将利用字段索引来尝试扫描最少量的日志事件，以查找 `requestId` 值为 123456 的事件

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

筛选命令中的匹配项和正则表达式

筛选条件支持使用正则表达式。您可以使用以下比较运算符 (=、!=、<、<=、>、>=) 和布尔运算符 (and、or 和 not)。

您可以使用关键字 `in` 来测试集合成员资格并检查数组中的元素。要检查数组中的元素，请将该数组放在 `in` 之后。您可以将布尔运算符 `not` 与 `in` 配合使用。您可以创建查询，它们使用 `in` 返回字段是字符串匹配项的录入事件。这些字段必须是完整字符串。例如，下面的代码片段显示了一个查询，它使用 `in` 返回字段 `logGroup` 是完整字符串 `example_group` 的录入事件。

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

您可以使用关键字短语 `like` 和 `not like` 以匹配子字符串。您可以使用正则表达式运算符 `=~` 以匹配子字符串。要使用 `like` 和 `not like` 匹配子字符串，请将您要匹配的子字符串括在单引号或双引号中。您可以将正则表达式模式与 `like` 和 `not like` 配合使用。要使用正则表达式运算符匹配子字符串，请将您要匹配的子字符串括在正斜杠中。下面的示例包含多个代码片段，它们展示您如何能够使用 `filter` 命令匹配子字符串。

示例：匹配子字符串

以下示例将返回 `f1` 包含单词 `Exception` 的录入事件。所有三个示例都区分大小写。

第一个示例使用 `like` 匹配子字符串。

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

第二个示例使用 `like` 和正则表达式模式匹配子字符串。

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

第三个示例使用正则表达式匹配子字符串。

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

示例：使用通配符匹配子字符串

您可以使用句点符号 (.) 作为正则表达式中的通配符来匹配子字符串。在以下示例中，查询返回 f1 的值以字符串 ServiceLog 开头的匹配项。

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

您可以在句点符号 (.) 之后放置星号，以创建一个返回尽可能多的匹配项的贪婪量词。例如，以下查询将返回 f1 的值不仅以字符串 ServiceLog 开头并且还包括字符串 ServiceLog 的匹配项。

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

可能的匹配项可以采用以下格式：

- ServiceLogSampleApiLogGroup
- SampleApiLogGroupServiceLog

示例：从匹配项中排除子字符串

下面的示例将显示一个查询，它将返回 f1 不包含单词 Exception 的多个录入事件。此示例区分大小写。

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

示例：使用不区分大小写的模式匹配子字符串

您可以使用 like 和正则表达式匹配不区分大小写的子字符串。在您要匹配的子字符串前放置以下参数 (?i)。下面的示例将显示一个查询，它将返回 f1 包含单词 Exception 或 exception 的多个录入事件。

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

filterIndex

使用 filterIndex 可以强制查询仅扫描在查询中指定的字段上编制索引的日志组，从而返回已编制索引的数据。对于已在此字段上编制索引的日志组，它会进一步优化查询，跳过没有任何日志事件包含索引字段的查询中指定的字段的日志组。这样可以进一步减少扫描量，因为它会尝试仅扫描这些日志

组中与此字段索引的查询中指定的值匹配的日志事件。有关字段索引和如何创建它们的更多信息，请参阅[创建字段索引以提高查询性能并减少扫描量](#)。

将 `filterIndex` 与索引字段结合使用，可以通过将实际搜索空间限制为具有字段索引的日志组和日志事件来帮助您高效地查询包含 PB 级日志数据的日志组。

例如，假设您已经在自己账户的某些日志组中为 `IPAddress` 创建了字段索引。然后，您可以创建以下查询，并选择查询账户中的所有日志组，以查找 `IPAddress` 字段包含值 `198.51.100.0` 的日志事件。

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

`filterIndex` 命令会使此查询尝试跳过所有未针对 `IPAddress` 编制索引的日志组。此外，在已编制索引的日志组中，查询会跳过具有 `IPAddress` 字段但未观察到该字段的值为 `198.51.100.0` 的日志事件。

使用 `IN` 运算符可以将结果扩展到索引字段的多个值中的任意一个。以下示例查找 `IPAddress` 字段包含值 `198.51.100.0` 或 `198.51.100.1` 的日志事件。

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

CloudWatch Logs 会为标准日志类中的所有日志组提供默认字段索引。默认字段索引可自动用于以下字段：

- `@logStream`
- `@aws.region`
- `@aws.account`
- `@source.log`
- `traceId`

默认字段索引是对您在策略中定义的任何自定义字段索引的补充。默认字段索引不计入[字段索引配额](#)。

filterIndex 与 filter 之比较

为了说明 filterIndex 和 filter 之间的区别，请考虑以下示例查询。假设您已经为四个日志组的 IPaddress 创建了字段索引，但没有为第五个日志组创建。以下使用 filterIndex 的查询将跳过扫描未为该字段编制索引的日志组。对于每个已编制索引的日志组，它会尝试仅扫描具有索引字段的日志事件，并且仅返回创建字段索引之后的结果。

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

相反，如果对相同的五个日志组使用 filter 而不是 filterIndex 进行查询，则查询不仅会尝试扫描已编制索引的日志组中包含该值的日志事件，还会扫描未编制索引的第五个日志组，并且会扫描该第五个日志组中的每个日志事件。

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

SOURCE

在使用 Amazon CLI 或 API 创建查询时，在查询中包含 SOURCE 是指定要包含在查询中的日志组的一种有效方法。SOURCE 命令仅在 Amazon CLI 和 API 中受支持，CloudWatch 控制台不支持。使用 CloudWatch 控制台启动查询时，您可以通过控制台界面指定日志组。

要使用 SOURCE 指定要查询的日志组，可以使用以下关键字：

- namePrefix 会针对名称以您指定的字符串开头的日志组运行查询。如果省略它，则会查询所有日志组。

列表中最多可以包含五个前缀。

- accountIdentifiers 会针对指定 Amazon 账户中的日志组运行查询。仅当您在监控账户中运行查询时，这才有效。如果省略此项，则默认为查询所有关联的源账户和当前的监控账户。有关跨账户可观测性的更多信息，请参阅 [CloudWatch 跨账户可观测性](#)。

列表中最多可以包含 20 个账户标识符。

- logGroupClass 会针对指定日志类（标准或不频繁访问）中的日志组运行查询。如果省略此项，则默认使用标准日志类。有关日志类的更多信息，请参阅 [日志类](#)。

由于您可以通过这种方式指定大量日志组进行查询，因此我们建议您仅在利用已创建的字段索引的查询中使用 SOURCE。有关为日志组中的字段编制索引的更多信息，请参阅[创建字段索引以提高查询性能并减少扫描量](#)

以下示例选择了账户中的所有日志组。如果这是一个监控账户，则会选择监控账户及其所有源账户中的所有日志组。如果日志组总数超过 10,000 个，则会显示一条错误消息，提示您使用其他日志组选择方法来减少日志组的数量。

```
SOURCE logGroups()
```

以下示例选择了 111122223333 源账户中的日志组。如果在 CloudWatch 跨账户可观测性中的监控账户中启动查询，则默认情况下会选择所有源账户和监控账户中的日志组。

```
SOURCE logGroups(accountIdentifiers:['111122223333'])
```

下一个示例根据名称前缀选择日志组。

```
SOURCE logGroups(namePrefix: ['namePrefix1', 'namePrefix2'])
```

以下示例选择了不频繁访问日志类中的所有日志组。如果不包含 class 标识符，则默认情况下，查询仅适用于标准日志类中的日志组。

```
SOURCE logGroups(class: ['INFREQUENT_ACCESS'])
```

下一个示例选择 111122223333 账户中以特定名称前缀开头且属于标准日志类的日志组。命令中未提及日志类，因为“标准”是默认的日志类值。

```
SOURCE logGroups(accountIdentifiers:['111122223333'], namePrefix: ['namePrefix1', 'namePrefix2'])
```

最后一个示例展示了如何将 SOURCE 命令与 start-query Amazon CLI 命令结合使用。

```
aws logs start-query
--region us-east-1
--start-time 1729728200
--end-time 1729728215
--query-string "SOURCE logGroups(namePrefix: ['Query']) | fields @message | limit 5"
```

模式

使用 `pattern` 自动将您的日志数据划分为不同模式。

模式是在日志字段中重复出现的共有的文本结构。您可以使用 `pattern` 来查看新出现的趋势、监控已知错误以及识别经常出现的或高成本的日志行。CloudWatch Logs Insights 还提供控制台体验，您可以使用它来查找并进一步分析日志事件中的模式。有关更多信息，请参阅 [模式分析](#)。

由于 `pattern` 命令会自动识别常见模式，因此您可以将其用作搜索和分析日志的起点。您还可以将 `pattern` 与 [filter](#)、[parse](#) 或 [sort](#) 命令结合使用，在更精细的查询中识别模式。

模式命令输入

`pattern` 命令需要以下输入之一：`@message` 字段、使用 [parse](#) 命令创建的提取字段或使用一个或多个 [字符串函数](#) 操作的字符串。

如果 CloudWatch Logs 无法推断动态令牌所代表的数据类型，则将其显示为 `<Token-##>`，`##` 表示此令牌与其他动态令牌相比在模式中出现的位置。

动态令牌的常见示例包括错误代码、IP 地址、时间戳以及请求 ID。

模式命令输出

`pattern` 命令将生成以下输出：

- `@pattern`：在日志事件字段中重复出现的共有的文本结构。在模式内变化的字段（例如请求 ID 或时间戳）由令牌表示。如果 CloudWatch Logs 可以确定动态令牌所代表的数据类型，它会将该令牌显示为 `<string-number>`。`###`是令牌所代表的数据类型的描述。`##`显示了与其他动态令牌相比，此令牌在模式中的出现位置。

CloudWatch Logs 根据分析包含该名称的日志事件的内容来分配名称的字符串部分。

如果 CloudWatch Logs 无法推断动态令牌所代表的数据类型，则将其显示为 `<Token-##>`，`##` 表示此令牌与其他动态令牌相比在模式中出现的位置。

例如，`[INFO] Request time: <Time-1> ms` 是日志消息 `[INFO] Request time: 327 ms` 的潜在输出。

- `@ratio`：选定时间段的日志事件与符合已确定模式的指定日志组的比率。例如，如果所选日志组和时间段中有一半的日志事件符合模式，则 `@ratio` 返回 `0.50`
- `@sampleCount`：选定时间段的日志事件与符合已确定模式的指定日志组的数量。

- `@severityLabel`：日志严重性或级别，表示日志中包含的信息类型。例如，`Error`、`Warning`、`Info` 或 `Debug`。

示例

以下命令识别选定时间范围内指定日志组中具有相似结构的日志，并按模式和计数对其进行分组

```
pattern @message
```

`pattern` 命令可以与 [filter](#) 命令结合使用

```
filter @message like /ERROR/  
| pattern @message
```

`pattern` 命令可以与 [parse](#) 和 [sort](#) 命令结合使用

```
filter @message like /ERROR/  
| parse @message 'Failed to do: *' as cause  
| pattern cause  
| sort @sampleCount asc
```

diff

将您请求的时间段内找到的日志事件与之前相同长度的时间段内的日志事件进行比较。这样，您就可以查找趋势并发现特定日志事件是否是新的。

在 `diff` 命令中添加修饰符以指定要比较的时间段：

- `diff` 将当前选定时间范围内的日志事件与紧接在前的时间范围内的日志事件进行比较。
- `diff previousDay` 将当前选定时间范围内的日志事件与前一天同一时间的日志事件进行比较。
- `diff previousWeek` 将当前选定时间范围内的日志事件与前一周同一时间的日志事件进行比较。
- `diff previousMonth` 将当前选定时间范围内的日志事件与上个月同一时间的日志事件进行比较。

有关更多信息，请参阅 [与之前的时间范围进行比较（差异）](#)。

parse

使用 `parse` 从日志字段中提取数据并创建可以在查询中处理的提取字段。**parse** 同时支持使用通配符和正则表达式的 `glob` 模式。有关正则表达式语法的信息，请参阅 [支持的正则表达式 \(regex \) 语法](#)。

您可以使用正则表达式解析嵌套的 JSON 字段。

示例：解析嵌套的 JSON 字段

该代码片段显示了如何解析在摄取过程中展平的 JSON 日志事件。

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

该代码片段显示了一个带有正则表达式的查询，其将提取 `fieldsA` 和 `fieldsB` 的值以创建提取字段 `fld` 和 `array`。

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

已命名的捕获组

配合正则表达式一起使用 **parse** 时，可以使用已命名的捕获组将模式捕获到字段中。语法为 `parse @message (?<Name>pattern)`。

下面的示例使用 VPC 流日志上的一个捕获组将 ENI 提取到名为“NetworkInterface”的字段中。

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

Note

JSON 日志事件在摄取过程中会被展平。目前，不支持使用 `glob` 表达式解析嵌套的 JSON 字段。您只能解析所含日志事件字段数不超过 200 的 JSON 日志事件。解析嵌套的 JSON 字段时，必须格式化查询中的正则表达式，使其与 JSON 日志事件的格式匹配。

解析命令的示例

使用 `glob` 表达式来从日志字段 `@message` 提取字段 `@user`、`@method` 和 `@latency`，并对于 `@method` 和 `@user` 的每个唯一组合返回平均延迟。

```
parse @message "user=*, method:*, latency := *" as @user,
```

```
@method, @latency | stats avg(@latency) by @method,
@user
```

使用正则表达式从日志字段 `@message` 提取字段 `@user2`、`@method2` 和 `@latency2`，并对于 `@method2` 和 `@user2` 的每个唯一组合返回平均延迟。

```
parse @message /user=(?<user2>.*?), method:(?<method2>.*?),
    latency := (?<latency2>.*?)/ | stats avg(latency2) by @method2,
    @user2
```

提取字段 **loggingTime**、**loggingType** 和 **loggingMessage**，筛选出包含 **ERROR** 或 **INFO** 字符串的日志事件，然后仅显示包含 **ERROR** 字符串的事件的 **loggingMessage** 和 **loggingType** 字段。

```

FIELDS @message
| PARSE @message "*" [*] "*" as loggingTime, loggingType, loggingMessage
| FILTER loggingType IN ["ERROR", "INFO"]
| DISPLAY loggingMessage, loggingType = "ERROR" as isError

```

排序

使用 `sort` 按指定字段依升序 (asc) 或降序 (desc) 顺序显示日志事件。您可以将其与 `limit` 命令一起使用，创建“前 N 个”或“后 N 个”查询。

该排序算法是自然排序的更新版本。如果按升序排序，则使用以下逻辑。

- 所有非数字值均位于所有数字值之前。数字值是仅包含数字的值，而不是数字和其他字符的混合。
- 对于非数字值，该算法将连续的数字字符和连续的字母字符分组为单独的块进行比较。它按 Unicode 值对非数字部分进行排序，并首先按长度对数字部分进行排序，然后按其数值进行排序。

有关 Unicode 顺序的更多信息，请参阅 [List of Unicode characters](#)。

例如，以下是按升序排序的结果。

```
!:>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 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_
```

[illegible]

如果按降序进行排序，则排序结果是相反的。

例如，以下针对 Amazon VPC 流日志的查询会发现主机间的前 15 个数据包传输。

```
stats sum(packets) as packetsTransferred by srcAddr, dstAddr
  | sort packetsTransferred desc
  | limit 15
```

stats

使用 `stats` 创建日志数据的可视化效果，例如条形图、折线图和堆叠面积图。这可以帮助您更有效地识别日志数据中的模式。CloudWatch Logs Insights 为使用 `stats` 函数以及一个或多个聚合函数的查询生成可视化效果。

例如，Route 53 日志组中的以下查询返回按查询类型显示 Route 53 记录每小时分布的可视化结果。

```
stats count(*) by queryType, bin(1h)
```

所有此类查询都可以生成条形图。如果查询使用 `bin()` 函数，按一个字段对数据随时间的变化进行分组，您还可以查看折线图和堆叠面积图。

`bin` 函数支持以下时间单位和缩写。对于所有包含多个字符的单位和缩写，支持添加复数形式。这样，`hr` 和 `hrs` 都可以指定小时数。

- 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

主题

- [可视化时间序列数据](#)
- [可视化按字段分组的日志数据](#)
- [在单个查询中使用多个 stats 命令](#)
- [用于统计数据的函数](#)

可视化时间序列数据

时间序列可视化适用于具有以下特征的查询：

- 此查询包含一个或多个聚合函数。有关更多信息，请参阅 [Aggregation Functions in the Stats Command](#)。
- 此查询使用 bin() 函数按一个字段对数据分组。

这些查询可以生成折线图、堆叠面积图、条形图和饼图。

示例

有关完整的教程，请参阅[the section called “教程：运行生成时间序列可视化的查询”](#)。

以下是更多适用于时间序列可视化的示例查询。

以下查询将生成 myfield1 字段的平均值的可视化（每 5 分钟创建一个数据点）。每个数据点都是前一个五分钟的日志中的 myfield1 值的平均值的聚合。

```
stats avg(myfield1) by bin(5m)
```

以下查询根据不同字段生成三个值的可视化（每 5 分钟创建一个数据点）。由于查询包含聚合函数并使用 bin() 作为分组字段，因此生成了可视化。

```
stats avg(myfield1), min(myfield2), max(myfield3) by bin(5m)
```

折线图和堆叠面积图限制

如果查询聚合日志条目信息但不使用 `bin()` 函数，则可以生成条形图。但是，这种查询无法生成折线图或堆叠面积图。有关这些查询类型的更多信息，请参阅[the section called “可视化按字段分组的日志数据”](#)。

可视化按字段分组的日志数据

您可以为使用 `stats` 函数以及一个或多个聚合函数的查询生成条形图。有关更多信息，请参阅[Aggregation Functions in the Stats Command](#)。

要查看可视化，请运行查询。接下来，选择 Visualization (可视化) 选项卡，选择 Line (折线图) 旁边的箭头，然后选择 Bar (条形图)。在条形图中，可视化限制为最多 100 个条形。

示例

有关完整的教程，请参阅[the section called “教程：运行生成按日志字段分组的可视化的查询”](#)。以下部分包括更多适用于按字段可视化的示例查询。

以下 VPC 流日志查询针对每个目标地址，查找每个会话所传输的平均字节数。

```
stats avg(bytes) by dstAddr
```

您还可以生成每个结果值包含多个条形图的图表。例如，以下 VPC 流日志查询针对每个目标地址，查找每个会话所传输的平均字节数和最大字节数。

```
stats avg(bytes), max(bytes) by dstAddr
```

以下查询为每个查询类型查找 Amazon Route 53 查询日志的数量。

```
stats count(*) by queryType
```

在单个查询中使用多个 stats 命令

您可以在单个查询中使用多达两个 `stats` 命令。这可让您对第一个聚合的输出执行额外的聚合。

示例：使用两个 **stats** 命令进行查询

例如，以下查询首先查找 5 分钟条柱中的总流量，然后计算这些 5 分钟条柱中的最高、最低和平均流量。

```
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
```

示例：将多个 stats 命令与其他函数（例如 **filter**、**fields**、**bin**）结合

您可以在单个查询中将两个 stats 命令与其他命令（例如 filter 和 fields）结合。例如，以下查询查找会话中不同 IP 地址的数量并按客户端平台查找会话数，筛选这些 IP 地址，最后找到每个客户端平台的会话请求平均值。

```
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
```

可以在带有多个 stats 命令的查询中使用 bin 和 dateceil 函数。例如，以下查询首先将消息组合成 5 分钟的块，然后将这些 5 分钟的块聚合为 10 分钟的块，并计算每个 10 分钟块内的最高、最低和平均流量。

```
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)
```

注释和限制

一个查询最多可以有两个 stats 命令。无法更改此配额。

如果您使用 sort 或 limit 命令，则它必须出现在第二个 stats 命令之后。如果它出现在第二个 stats 命令之前，则查询无效。

当一个查询有两个 stats 命令时，查询的部分结果要等到第一个 stats 聚合完成后才会开始显示。

在单个查询的第二个 stats 命令中，您只能引用第一个 stats 命令中定义的字段。例如，以下查询无效，因为 @message 字段在第一个 stats 聚合后将不可用。

```
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
```

在第一个 stats 命令之后引用的任何字段都必须在此第一个 stats 命令中定义。

```
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

bin 函数始终隐式使用 @timestamp 字段。这意味着，如果未使用第一个 stats 命令传播 timestamp 字段，就无法在第二个 stats 命令中使用 bin。例如，以下查询无效。

```
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
```

相反，在第一个 stats 命令中定义 @timestamp 字段，然后就可以在第二个 stats 命令中结合 dateceil 使用该字段，如下例所示。

```
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)
```

用于统计数据的函数

CloudWatch Logs Insights 支持统计聚合函数和统计非聚合函数。

在 stats 命令中使用统计聚合函数，并将其用作其他函数的参数。

函数	结果类型	描述
avg(fieldName: NumericLogField)	数字	指定的字段中值的平均值。
count()	数字	计算日志事件的数量。count() (或 count(*)) 对查询返回的所有事件进行计数，

函数	结果类型	描述
<code>count(fieldName: LogField)</code>		而 <code>count(fieldName)</code> 对包含所指定字段名称的所有记录进行计数。
<code>count_distinct(fieldName: LogField)</code>	数字	返回字段的唯一值的数量。如果字段具有非常高的基数（包含许多唯一值），则 <code>count_distinct</code> 返回的值只是一个近似值。
<code>max(fieldName: LogField)</code>	LogFieldValue	所查询的日志中此日志字段的值的最大值。
<code>min(fieldName: LogField)</code>	LogFieldValue	所查询的日志中此日志字段的值的最小值。
<code>pct(fieldName: LogFieldValue, percent: number)</code>	LogFieldValue	百分位数指示某个值在数据集中的相对位置。例如， <code>pct(@duration, 95)</code> 返回 <code>@duration</code> 值，95% 的 <code>@duration</code> 值低于此值，5% 的值高于此值。
<code>stddev(fieldName: NumericLogField)</code>	数字	指定的字段中值的标准偏差。
<code>sum(fieldName: NumericLogField)</code>	数字	指定的字段中值的总和。

统计非聚合函数

在 `stats` 命令中使用非聚合函数并将其用作其他函数的参数。

函数	结果类型	描述
<code>earliest(fieldName: LogField)</code>	LogField	从查询的日志中具有最早时间戳的日志事件返回 <code>fieldName</code> 的值。
<code>latest(fieldName: LogField)</code>	LogField	从查询的日志中具有最晚时间戳的日志事件返回 <code>fieldName</code> 的值。

函数	结果类型	描述
<code>sortsFirst(fieldName: LogField)</code>	LogField	返回在查询的日志中排在第一位的 <code>fieldName</code> 的值。
<code>sortsLast(fieldName: LogField)</code>	LogField	返回在查询的日志中排在最后一位的 <code>fieldName</code> 的值。

limit

使用 `limit` 指定您希望查询返回的日志事件数。如果省略 `limit`，则查询将在结果中返回多达 10,000 个日志事件。

例如，下面的示例仅返回最近 25 个日志事件。

```
fields @timestamp, @message | sort @timestamp desc | limit 25
```

dedup

使用 `dedup` 根据您指定的字段中的特定值删除重复的结果。您可以将 `dedup` 与一个或多个字段配合使用。如果为一个字段指定 `dedup`，则会仅为该字段的每个唯一值返回一个日志事件。如果指定多个字段，则会为这些字段的每个唯一值组合返回一个日志事件。

重复项会根据排序顺序被丢弃，只保留排序顺序中的第一个结果。我们建议您在 `dedup` 执行命令之前对结果进行排序。如果在运行 `dedup` 之前未对结果进行排序，则会运用使用 `@timestamp` 时的默认降序排序顺序。

在评估中，NULL 值不被视为重复值。任何指定字段的值为 NULL 值的日志事件都将被保留。要消除具有 NULL 值的字段，请运用使用 `isPresent(field)` 函数的 **filter**。

在 `dedup` 命令之后可以在查询中使用的唯一查询命令是 `limit`。

示例：仅查看名为 **server** 的字段的每个唯一值的最新日志事件

以下示例仅显示 `server` 每个唯一值的最新事件的 `timestamp`、`server`、`severity` 和 `message` 字段。

```
fields @timestamp, server, severity, message
| sort @timestamp desc
| dedup server
```

有关 CloudWatch Logs Insights 查询的更多示例，请参阅 [常规查询](#)。

unmask

使用 unmask 显示由于数据保护策略而屏蔽部分内容的日志事件的所有内容。要使用此命令，必须具备 logs:Unmask 权限。

有关日志组中数据保护的更多信息，请参阅 [通过屏蔽帮助保护敏感的日志数据](#)。

unnest

使用 unnest 可以对输入列表进行扁平化以生成多条记录，列表中的每个元素都有一条记录。根据字段包含的项目数量，此命令会丢弃当前记录并生成新记录。每条记录都包含 unnested_field，它代表一个项目。其他所有字段均来自原始记录。

unnest 的输入是 LIST，它来自 jsonParse 函数。有关更多信息，请参阅[结构类型](#)。任何其他类型（例如，MAP、String 和 numbers）都被视为 unnest 中包含一个项目的列表。

命令结构

以下示例描述了此命令的格式。

```
unnest field into unnested_field
```

示例查询

以下示例解析了 JSON 对象字符串并展开字段事件列表。

```
fields jsonParse(@message) as json_message
| unnest json_message.events into event
| display event.name
```

此示例查询的日志事件可能是一个 JSON 字符串，如下所示：

```
{
  "events": [
    {
      "name": "exception"
    },
    {
      "name": "user action"
    }
  ]
}
```

```
}
```

在本例中，示例查询会在查询结果中生成两条记录，一条记录的 `event.name` 为 `exception`，另一条记录的 `event.name` 为 `user action`。

示例查询

以下示例会对列表进行扁平化，然后筛选出项目。

```
fields jsonParse(@message) as js
| unnest js.accounts into account
| filter account.type = "internal"
```

示例查询

以下示例会对列表进行扁平化以进行聚合。

```
fields jsonParse(trimmedData) as accounts
| unnest accounts into account
| stats sum(account.droppedSpans) as n by account.accountId
| sort n desc
| limit 10
```

布尔值、比较、数值、日期时间和其他函数

CloudWatch Logs Insights 在查询中支持许多其他操作和函数，如以下各部分所述。

主题

- [算术运算符](#)
- [布尔运算符](#)
- [比较运算符](#)
- [数值运算符](#)
- [结构类型](#)
- [日期时间函数](#)
- [常规函数](#)
- [JSON 函数](#)
- [IP 地址字符串函数](#)
- [字符串函数](#)

算术运算符

算术运算接受数值数据类型作为参数并返回数值结果。在 `filter` 和 `fields` 命令中使用算术运算并将其用作其他函数的参数。

操作	描述
$a + b$	加
$a - b$	减
$a * b$	乘
a / b	除
$a ^ b$	幂 ($2 ^ 3$ 返回 8)
$a \% b$	余额或模数 ($10 \% 3$ 返回 1)

布尔运算符

使用布尔运算符 **and**、**or** 和 **not**。

Note

仅在返回 TRUE 或 FALSE 值的函数中使用布尔运算符。

比较运算符

比较运算接受所有数据类型作为参数，并返回布尔值结果。在 `filter` 命令中使用比较运算并将其用作其他函数的参数。

运算符	描述
<code>=</code>	Equal
<code>!=</code>	Not equal
<code><</code>	Less than

运算符	描述
>	Greater than
<=	小于或等于
>=	大于或等于

数值运算符

数值运算接受数值数据类型作为参数并返回数值结果。在 `filter` 和 `fields` 命令中使用数值运算并将其用作其他函数的参数。

操作	结果类型	描述
<code>abs(a: number)</code>	number	绝对值
<code>ceil(a: number)</code>	数字	舍入到上限 (大于 a 的的最小整数)
<code>floor(a: number)</code>	数字	舍入到下限 (小于 a 的值的最大整数)
<code>greatest(a: number, ...numbers: number[])</code>	数字	返回最大值
<code>least(a: number, ...numbers: number[])</code>	数字	返回最小值
<code>log(a: number)</code>	数字	自然对数
<code>sqrt(a: number)</code>	数字	平方根

结构类型

映射或列表是 CloudWatch Logs Insights 中的一种结构类型，允许您访问和使用属性进行查询。

示例：获取映射或列表

使用 `jsonParse` 可以将 json 字符串字段解析为地图或列表。

```
fields jsonParse(@message) as json_message
```

示例：访问属性

使用点访问运算符 (`map.attribute`) 可以访问映射中的项。如果映射中的某属性包含特殊字符，请使用反引号将属性名称括起来 (`map.attributes.`special.char``)。

```
fields jsonParse(@message) as json_message  
| stats count() by json_message.status_code
```

使用方括号访问运算符 (`list[index]`) 可以检索列表中特定位置的项。

```
fields jsonParse(@message) as json_message  
| filter json_message.users[1].action = "PutData"
```

当键名中出现特殊字符时，请用反引号 (```) 将特殊字符括起来。

```
fields jsonParse(@message) as json_message  
| filter json_message.`user.id` = "123"
```

示例：空结果

对于字符串、数字和日期时间函数，映射和列表会被视为空值。

```
fields jsonParse(@message) as json_message  
| display toupper(json_message)
```

将映射和列表与任何其他字段进行比较会得出 `false`。

Note

不支持在 `dedup`、`pattern`、`sort` 和 `stats` 中使用映射和列表。

日期时间函数

日期时间函数

在 `fields` 和 `filter` 命令中使用日期时间函数并将其用作其他函数的参数。使用这些函数为使用聚合函数的查询创建时间存储桶。使用由数字和下列项之一组成的时间段：

- `ms` 代表毫秒
- `s` 代表秒
- `m` 代表分钟
- `h` 代表小时

例如，`10m` 为 10 分钟，`1h` 为 1 小时。

 Note

为日期时间函数使用最合适的时间单位。CloudWatch Logs 会根据您选择的时间单位限制您的请求。例如，对于使用 `s` 的任何请求，它将最大值限制为 60。因此，如果指定 `bin(300s)`，CloudWatch Logs 实际上会将其实现为 60 秒，因为 60 是一分钟的秒数，因此 CloudWatch Logs 不会将高于 60 的数字与 `s` 一起使用。要创建 5 分钟的存储桶，请改用 `bin(5m)`。
`ms` 的上限为 1000，`s` 和 `m` 的上限为 60，`h` 的上限为 24。

下表包含可在查询命令中使用的不同日期时间函数列表。该表列出了每个函数的结果类型，并包含对每个函数的描述。

 Tip

在创建查询命令时，您可以使用时间间隔选择器选择要查询的时间段。例如，您可以设置 5 到 30 分钟的时间间隔；1、3 和 12 小时间隔；或者自定义时间范围。您还可以设置特定日期之间的时间段。

函数	结果类型	描述
<code>bin(period: Period)</code>	Timestamp	将 <code>@timestamp</code> 的值四舍五入到指定的时间段，然后截断。例如， <code>bin(5m)</code> 将 <code>@timestamp</code> 的值四舍五入到最近的 5 分钟。

函数	结果类型	描述
		您可以使用它将某个查询中的多个日志条目分为一组。以下示例返回每小时的异常数量。 <pre>filter @message like /Exception/ stats count(*) as exceptionCount by bin(1h) sort exceptionCount desc</pre> bin 函数支持以下时间单位和缩写。对于所有包含多个字符的单位和缩写，支持添加复数形式。这样，hr 和 hrs 都可以指定小时数。 • 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
datefloor(timestamp: Timestamp, period: Period)	Timestamp	将时间戳截断到指定的时间段。例如，datefloor(@timestamp, 1h) 将 @timestamp 的所有值截断至小时底部。
dateceil(timestamp : Timestamp, period: Period)	Timestamp	将时间戳向上舍入到指定的时间段，然后截断。例如，dateceil(@timestamp, 1h) 将 @timestamp 的所有值截断至小时顶部。
fromMillis(fieldName: number)	Timestamp	将输入字段解释为自 Unix epoch 以来的毫秒数并将其转换为时间戳。

函数	结果类型	描述
<code>toMillis(fieldName: Timestamp)</code>	数字	将在命名字段中找到的时间戳转换为表示自 Unix epoch 以来毫秒数的数字。例如，<code>toMillis(@timestamp)</code> 将时间戳 <code>2022-01-14T13:18:031.000-08:00</code> 转换为 <code>1642195111000</code>。
<code>now()</code>	数字	返回查询处理开始的时间，以 epoch 秒为单位。此函数不接受任何参数。 您可以使用它根据当前时间筛选查询结果。 例如，以下查询将返回过去两小时内的所有 4xx 错误： <pre> parse @message "Status Code: *;" as statusCode\n filter statusCode >= 400 and statusCode <= 499 \n filter toMillis(@timestamp) >= (now() * 1000 - 7200000) </pre> 以下示例会返回过去五小时内所有包含 <code>error</code> 或 <code>failure</code> 字样的日志条目 <pre> fields @timestamp, @message filter @message like /(?(i)(error failure))/ filter toMillis(@timestamp) >= (now() * 1000 - 18000000) </pre>

Note

目前，CloudWatch Logs Insights 不支持使用人类可读的时间戳筛选日志。

常规函数

常见函数

在 `fields` 和 `filter` 命令中使用常规函数并将其用作其他函数的参数。

函数	结果类型	描述
<code>ispresent(fieldName: LogField)</code>	布尔值	如果字段存在，则返回 <code>true</code>
<code>coalesce(fieldName: LogField, ...fieldNames: LogField[])</code>	LogField	返回列表中的第一个非 <code>null</code> 值

JSON 函数

JSON 函数

在 `fields` 和 `filter` 命令中使用 JSON 函数并将其用作其他函数的参数。

函数	结果类型	描述
<code>jsonParse(fieldName: string)</code>	映射 列表 空	当输入为 JSON 对象的字符串表示形式或 JSON 数组时，返回映射或列表。如果输入不是其中一种表示形式，则返回空值。
<code>jsonStringify(fieldName: Map List)</code>	字符串	从映射或列表数据中返回 JSON 字符串。

IP 地址字符串函数

IP 地址字符串函数

在 `filter` 和 `fields` 命令中使用 IP 地址字符串函数并将其用作其他函数的参数。

函数	结果类型	描述
<code>isValidIp(fieldName: string)</code>	布尔值	如果字段是有效的 IPv4 或 IPv6 地址，则返回 <code>true</code> 。
<code>isValidIPv4(fieldName: string)</code>	布尔值	如果字段是有效的 IPv4 地址，则返回 <code>true</code> 。
<code>isValidIPv6(fieldName: string)</code>	布尔值	如果字段是有效的 IPv6 地址，则返回 <code>true</code> 。
<code>isIpInSubnet(fieldName: string, subnet: string)</code>	布尔值	如果字段是指定的 v4 或 v6 子网中有效的 IPv4 或 IPv6 地址，则返回 <code>true</code> 。指定子网时，请使用 CIDR 表示法（例如 <code>192.0.2.0/24</code> 或 <code>2001:db8::/32</code> ），其中 <code>192.0.2.0</code> 或 <code>2001:db8::</code> 是 CIDR 块的起点。
<code>isIPv4InSubnet(fieldName: string, subnet: string)</code>	布尔值	如果字段是指定的 v4 子网中有效的 IPv4 地址，则返回 <code>true</code> 。指定子网时，请使用 CIDR 表示法（例如 <code>192.0.2.0/24</code> ），其中 <code>192.0.2.0</code> 是 CIDR 块的起点。
<code>isIPv6InSubnet(fieldName: string, subnet: string)</code>	布尔值	如果字段是指定的 v6 子网中有效的 IPv6 地址，则返回 <code>true</code> 。指定子网时，请使用 CIDR 表示法（例如 <code>2001:db8::/32</code> ），其中 <code>2001:db8::</code> 是 CIDR 块的起点。

字符串函数

字符串函数

在 `fields` 和 `filter` 命令中使用字符串函数并将其用作其他函数的参数。

函数	结果类型	描述
<code>isempty(fieldName: string)</code>	数字	如果字段缺失或为空字符串，则返回 1。

函数	结果类型	描述
<code>isblank(fieldName: string)</code>	数字	如果字段缺失或为空字符串，或只包含空格，则返回 1。
<code>concat(str: string, ...strings: string[])</code>	字符串	连结字符串。
<code>ltrim(str: string)</code> <code>ltrim(str: string, trimChars: string)</code>	字符串	如果函数没有第二个参数，它将删除字符串左侧的空格。如果函数有第二个字符串参数，它将不会删除空格。相反，它会从 <code>str</code> 左侧删除 <code>trimChars</code> 中的字符。例如， <code>ltrim("xyZxyfooxyZ", "xyZ")</code> 将返回 <code>"fooxyZ"</code> 。
<code>rtrim(str: string)</code> <code>rtrim(str: string, trimChars: string)</code>	字符串	如果函数没有第二个参数，它将删除字符串右侧的空格。如果函数有第二个字符串参数，它将不会删除空格。相反，它会从 <code>str</code> 右侧删除 <code>trimChars</code> 字符。例如， <code>rtrim("xyZfooxyxyZ", "xyZ")</code> 将返回 <code>"xyZfoo"</code> 。
<code>trim(str: string)</code> <code>trim(str: string, trimChars: string)</code>	字符串	如果函数没有第二个参数，它将删除字符串两端的空格。如果函数有第二个字符串参数，它将不会删除空格。相反，它会从 <code>str</code> 两端删除 <code>trimChars</code> 字符。例如， <code>trim("xyZxyfooxyxyZ", "xyZ")</code> 将返回 <code>"foo"</code> 。

函数	结果类型	描述
<code>strlen(str: string)</code>	数字	返回 Unicode 代码点中字符串的长度。
<code>toupper(str: string)</code>	字符串	将字符串转换为大写。
<code>tolower(str: string)</code>	字符串	将字符串转换为小写。
<code>substr(str: string, startIndex: number)</code> <code>substr(str: string, startIndex: number, length: number)</code>	字符串	返回从由数值参数指定的索引到字符串末尾的子字符串。如果该函数具有二个参数，它包含要检索的子字符串的长度。例如， <code>substr("xyzfooxyz", 3, 3)</code> 将返回 "foo"。
<code>replace(fieldName: string, searchValue: string, replaceValue: string)</code>	字符串	将 <code>fieldName: string</code> 中出现的所有 <code>searchValue</code> 替换为 <code>replaceValue</code> 。 例如，函数 <code>replace(logGroup, "smoke_test", "Smoke")</code> 搜索录入事件，其中字段 <code>logGroup</code> 包含字符串值 <code>smoke_test</code>，并使用字符串 <code>Smoke</code> 替换该值。
<code>strcontains(str: string, searchValue: string)</code>	数字	如果 <code>str</code> 包含 <code>searchValue</code> ，则返回 1，否则返回 0。

包含特殊字符的字段

如果字段包含除 @ 符号或句点 (.) 之外的非字母数字字符，则必须用反引号字符 (` ) 将该字段括起来。例如，日志字段 `foo-bar` 必须括在反引号中 (``foo-bar``)，因为它包含连字符 (-)，这是非字母数字字符。

在查询中使用别名和注释

创建包含别名的查询。使用别名重命名日志字段或将值提取到字段中时重命名。使用关键字 `as` 为日志字段或结果提供别名。可以在查询中使用多个别名。可以在以下命令中使用别名：

- `fields`
- `parse`
- `sort`
- `stats`

以下示例演示了如何创建包含别名的查询。

示例：

查询中在 `fields` 命令中包含别名。

```
fields @timestamp, @message, accountId as ID
| sort @timestamp desc
| limit 20
```

查询将返回字段 `@timestamp`、`@message` 和 `accountId` 的值。结果按降序排序且限制在 20 个。`accountId` 的值在别名 `ID` 下列出。

示例：

查询在 `sort` 和 `stats` 命令中包含别名。

```
stats count(*) by duration as time
| sort time desc
```

查询计算字段 `duration` 出现在日志组中的次数，然后按降序对结果进行排序。`duration` 的值在别名 `time` 下列出。

使用注释

CloudWatch Logs Insights 支持查询中的备注。使用哈希字符 (`#`) 分隔备注。您可以使用备注忽略查询或文档查询中的行。

示例：查询

运行以下查询时，将忽略第二行。

```
fields @timestamp, @message, accountId
# | filter accountId not like "7983124201998"
| sort @timestamp desc
| limit 20
```

开始使用 Logs Insights QL : 查询教程

以下各部分包括示例查询教程，可帮助您开始使用 Logs Insights QL。

主题

- [教程：运行和修改示例查询](#)
- [教程：使用聚合函数运行查询](#)
- [教程：运行生成按日志字段分组的可视化的查询](#)
- [教程：运行生成时间序列可视化的查询](#)

教程：运行和修改示例查询

以下教程可帮助您开始使用 CloudWatch Logs Insights。您可以在 Logs Insights QL 中运行示例查询，然后了解如何修改并重新运行它。

要运行查询，CloudWatch Logs 中必须已存储有日志。如果您已在使用 CloudWatch Logs 并设置了日志组和日志流，那么您已准备就绪，可以开始使用它了。如果您使用服务（如 Amazon CloudTrail、Amazon Route 53 或 Amazon VPC）并且您已将日志设置为从这些服务转到 CloudWatch Logs，则您可能已有日志。有关将事件发送到 CloudWatch Logs 的更多信息，请参阅 [开始使用 CloudWatch Logs](#)。

CloudWatch Logs Insights 中的查询返回日志事件中的一组字段，或返回对日志事件执行的数学聚合或其他运算的结果。本教程演示了一个查询，该查询返回您的日志事件列表。

运行示例查询

运行 CloudWatch Logs Insights 示例查询

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Logs（日志），然后选择 Logs Insights（日志洞察）。

在 Logs Insights 页面上，查询编辑器包含 Logs Insights QL 中的一个默认查询，它将返回 20 个最近的日志事件。

3. 在 Select log group(s) (选择日志组) 在下拉菜单中，选择要查询的一个或多个日志组。

如果这是 CloudWatch 跨账户可观察性中的监控账户，则您可以在源账户和监控账户中选择日志组。单个查询可以同时查询来自不同账户的日志。

您可以按日志组名称、账户 ID 或账户标签筛选日志组。

当您选择标准日志类中的日志组时，CloudWatch Logs Insights 会自动检测该组中的数据字段。要查看这些搜索到的字段，请选择页面右上方的 Fields (字段) 菜单。

Note

仅标准日志类中的日志组支持已发现字段。有关日志类的更多信息，请参阅 [日志类](#)。

4. (可选) 使用时间间隔选择器选择要查询的时间段。

您可以选择 5 到 30 分钟的间隔；1、3 和 12 小时间隔；或者自定义时间范围。

5. 选择 Run (运行) 以查看结果。

在本教程中，结果包括 20 个最近添加的日志事件。

CloudWatch Logs 会显示此日志组中的日志事件随时间变化的条形图。该条形图显示日志组中与查询和时间范围匹配的事件分布情况，而不仅仅是表中显示的事件。

6. 要查看返回的日志事件的所有字段，请选择编号事件左侧的三角形下拉图标。

修改示例查询

在本教程中，您将修改示例查询以显示 50 个最新的日志事件。

如果您尚未运行上一教程，请立即运行。本教程开始于前一教程结束的位置。

Note

随 CloudWatch Logs Insights 提供的一些示例查询使用 head 或 tail 命令，而非 limit。这些命令现正被弃用并已替换为 limit。在写入的所有查询中使用 limit 而非 head 或 tail。

修改 CloudWatch Logs Insights 示例查询

1. 在查询编辑器中，将 20 更改为 50，然后选择 Run (运行)。

显示新查询的结果。假设在默认时间范围内日志组中有足够的数据，则现在将列出 50 个日志事件。

2. (可选) 您可以保存已创建的查询。要保存此查询，请选择 Save (保存)。有关更多信息，请参阅 [保存并重新运行 CloudWatch Logs Insights 查询](#)。

将筛选命令添加到示例查询

本教程说明如何在查询编辑器中对查询进行更有效的更改。在本教程中，您将基于检索的日志事件中的某个字段筛选上一个查询的结果。

如果您尚未运行前面的教程，请立即运行。本教程开始于前一教程结束的位置。

将筛选命令添加到前一查询

1. 确定要筛选的字段。要查看 CloudWatch Logs 在过去 15 分钟内所选日志组中包含的日志事件内检测到的最常见字段，以及每个字段显示的那些日志事件的百分比，请选择页面右侧的 Fields (字段)。

要查看特定日志事件中包含的字段，请选择该行左侧的图标。

awsRegion 字段可能会显示在您的日志事件中，具体取决于日志中的事件。对于本教程的其余部分，我们使用 awsRegion 作为筛选字段，但如果该字段不可用，则您可以使用其他字段。

2. 在查询编辑器框中，将光标放在 50 之后，然后按 Enter。
3. 在新行上，首先输入 | (竖线字符) 和一个空格。必须用竖线字符分隔 CloudWatch Logs Insights 查询中的命令。
4. 输入 **filter awsRegion="us-east-1"**。
5. 选择运行。

此查询再次运行，现在将显示与新筛选器匹配的 50 个最新结果。

如果您筛选不同的字段并获得错误结果，则您可能需要对字段名称进行转义。如果字段名称包含非字母数字字符，则必须在字段名称前后放入反引号字符 (`)，例如 **``error-code`="102"`**。

必须将反引号字符用于包含非字母数字字符的字段名称，但不能用于值。值始终包含在引号 (") 中。

Logs Insights QL 包含强大的查询能力，包括若干命令和对于正则表达式、数学和统计运算的支持。有关更多信息，请参阅 [CloudWatch Logs Insights 语法查询语法](#)。

教程：使用聚合函数运行查询

您可以通过 `stats` 命令使用聚合函数并将其用作其他函数的参数。在本教程中，您会运行一个查询命令，该命令将计算包含指定字段的录入事件的数量。查询命令返回按指定字段的一个值或多个值分组的总计数。有关聚合函数的更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[支持的运算和函数](#)。

使用聚合函数运行查询

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Logs (日志)，然后选择 Logs Insights (日志洞察)。
3. 确认已选中 Logs Insights QL 选项卡。
4. 在 Select log group(s) (选择日志组) 在下拉菜单中，选择要查询的一个或多个日志组。

如果这是 CloudWatch 跨账户可观察性中的监控账户，则您可以在源账户和监控账户中选择日志组。单个查询可以同时查询来自不同账户的日志。

您可以按日志组名称、账户 ID 或账户标签筛选日志组。

当您选择一个日志组时，如果它是标准类日志组，CloudWatch Logs Insights 会自动检测该日志组中的数据字段。要查看这些搜索到的字段，请选择页面右上方的 Fields (字段) 菜单。

5. 在查询编辑器中删除原定设置查询，然后输入以下命令：

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

6. 使用 Fields (字段) 菜单中已发现的字段替换 *fieldName*。

Fields (字段) 菜单位于页面的右上角，显示 CloudWatch Logs Insights 在日志组中检测到的所有已发现的字段。

7. 选择 Run (运行) 以查看查询结果。

查询结果会显示日志组中与查询命令匹配的记录数以及按指定字段一个值或多个值分组的总计数。

教程：运行生成按日志字段分组的可视化的查询

当您运行的查询使用 `stats` 函数按日志条目中一个或多个字段的值来分组所返回的结果时，您可以使用条形图、饼图、折线图或堆叠面积图来查看结果。这可帮助您更有效地将日志中的趋势可视化。

运行查询进行可视化

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Logs (日志)，然后选择 Logs Insights (日志洞察)。
3. 在 Select log group(s) (选择日志组) 在下拉菜单中，选择要查询的一个或多个日志组。

如果这是 CloudWatch 跨账户可观察性中的监控账户，则您可以在源账户和监控账户中选择日志组。单个查询可以同时查询来自不同账户的日志。

您可以按日志组名称、账户 ID 或账户标签筛选日志组。

4. 在查询编辑器中，删除当前内容，然后输入以下 `stats` 函数并选择 Run query (运行查询)。

```
stats count(*) by @logStream
| limit 100
```

对于每个日志流，结果显示日志组中的日志事件数量。结果限制为 100 行。

5. 选择 Visualization (可视化) 选项卡。
6. 选择 Line (折线图) 旁边的箭头，然后选择 Bar (条形图)。

此时将出现条形图，其中显示日志组中各个日志流的条形图。

教程：运行生成时间序列可视化的查询

当您运行使用 `bin()` 函数按时间段对返回的结果进行分组的查询时，您可以使用折线图、堆叠面积图、饼图或条形图来查看结果。这有助于您更有效地将日志事件随时间变化的趋势可视化。

运行查询进行可视化

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Logs (日志)，然后选择 Logs Insights (日志洞察)。
3. 确认已选中 Logs Insights QL 选项卡。
4. 在 Select log group(s) (选择日志组) 在下拉菜单中，选择要查询的一个或多个日志组。

如果这是 CloudWatch 跨账户可观察性中的监控账户，则您可以在源账户和监控账户中选择日志组。单个查询可以同时查询来自不同账户的日志。

您可以按日志组名称、账户 ID 或账户标签筛选日志组。

5. 在查询编辑器中，删除当前内容，然后输入以下 stats 函数并选择 Run query (运行查询)。

```
stats count(*) by bin(30s)
```

结果显示日志组中由 CloudWatch Logs 每 30 秒时间段接收的日志事件的数量。

6. 选择 Visualization (可视化) 选项卡。

结果显示为折线图。要切换到条形图、饼图或堆叠面积图，请在图形左上角选择 Line (折线图) 旁边的箭头。

示例查询

本部分包含常用的查询命令列表，您可以在 [CloudWatch 控制台](#) 中运行这些命令。有关如何运行查询命令的信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[教程：运行和修改示例查询](#)。

有关查询语法的更多信息，请参阅 [CloudWatch Logs Insights 语法查询语法](#)。

主题

- [常规查询](#)
- [Lambda 日志的查询](#)
- [Amazon VPC 流日志的查询](#)
- [Route 53 日志的查询](#)
- [CloudTrail 日志的查询](#)
- [对 Amazon API Gateway 的查询](#)
- [NAT 网关的查询](#)
- [查询 Apache 服务器日志](#)
- [针对 Amazon EventBridge 的查询](#)
- [解析命令的示例](#)

常规查询

查找 25 个最近添加的日志事件。

```
fields @timestamp, @message | sort @timestamp desc | limit 25
```

获取每小时异常数量的列表。

```
filter @message like /Exception/  
  | stats count(*) as exceptionCount by bin(1h)  
  | sort exceptionCount desc
```

获取非异常的日志事件的列表。

```
fields @message | filter @message not like /Exception/
```

获取 **server** 字段每个唯一值的最新日志事件。

```
fields @timestamp, server, severity, message  
  | sort @timestamp asc  
  | dedup server
```

针对每个 **severity** 类型获取 **server** 字段每个唯一值的最新日志事件。

```
fields @timestamp, server, severity, message  
  | sort @timestamp desc  
  | dedup server, severity
```

Lambda 日志的查询

确定超额配置的内存量。

```
filter @type = "REPORT"  
  | stats max(@memorySize / 1000 / 1000) as provisionedMemoryMB,  
    min(@maxMemoryUsed / 1000 / 1000) as smallestMemoryRequestMB,  
    avg(@maxMemoryUsed / 1000 / 1000) as avgMemoryUsedMB,  
    max(@maxMemoryUsed / 1000 / 1000) as maxMemoryUsedMB,  
    provisionedMemoryMB - maxMemoryUsedMB as overProvisionedMB
```

创建延迟报告。

```
filter @type = "REPORT" |
  stats avg(@duration), max(@duration), min(@duration) by bin(5m)
```

搜索慢速函数调用，并消除可能因重试或客户端代码而产生的重复请求。在此查询中，**@duration** 以毫秒为单位。

```
fields @timestamp, @requestId, @message, @logStream
| filter @type = "REPORT" and @duration > 1000
| sort @timestamp desc
| dedup @requestId
| limit 20
```

Amazon VPC 流日志的查询

查找跨主机的前 15 个数据包传输：

```
stats sum(packets) as packetsTransferred by srcAddr, dstAddr
| sort packetsTransferred desc
| limit 15
```

查找给定子网上传输字节数最多的 15 个主机。

```
filter isIpv4InSubnet(srcAddr, "192.0.2.0/24")
| stats sum(bytes) as bytesTransferred by dstAddr
| sort bytesTransferred desc
| limit 15
```

查找使用 UDP 作为数据传输协议的 IP 地址。

```
filter protocol=17 | stats count(*) by srcAddr
```

在捕获时段内查找跳过流记录的 IP 地址。

```
filter logStatus="SKIPDATA"
| stats count(*) by bin(1h) as t
| sort t
```

为每个连接查找一条记录，以帮助解决网络连接问题。

```
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
```

Route 53 日志的查询

查找每小时每种查询类型的记录分布。

```
stats count(*) by queryType, bin(1h)
```

查找具有最高请求数的 10 个 DNS 解析程序。

```
stats count(*) as numRequests by resolverIp
| sort numRequests desc
| limit 10
```

按服务器未能完成 DNS 请求的域和子域查找记录数。

```
filter responseCode="SERVFAIL" | stats count(*) by queryName
```

CloudTrail 日志的查询

查找每项服务、事件类型和 Amazon 区域的日志条目数。

```
stats count(*) by eventSource, eventName, awsRegion
```

查找给定 Amazon 区域中已启动或已停止的 Amazon EC2 主机。

```
filter (eventName="StartInstances" or eventName="StopInstances") and awsRegion="us-east-2"
```

查找新建的 IAM 用户的 Amazon 区域、用户名和 ARN。

```
filter eventName="CreateUser"
```



```
| fields awsRegion, requestParameters.userName, responseElements.user.arn
```

查找在调用 API **UpdateTrail** 时发生异常的记录数。

```
filter eventName="UpdateTrail" and ispresent(errorCode)
| stats count(*) by errorCode, errorMessage
```

查找使用 TLS 1.0 或 1.1 的日志条目

```
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
```

查找使用 TLS 1.0 或 1.1 的服务调用数

```
filter tlsDetails.tlsVersion in [ "TLSv1", "TLSv1.1" ]
| stats count(*) as numOutdatedTlsCalls by eventSource
| sort numOutdatedTlsCalls desc
```

对 Amazon API Gateway 的查询

查找最近 10 个 4XX 错误

```
fields @timestamp, status, ip, path, httpMethod
| filter status>=400 and status<=499
| sort @timestamp desc
| limit 10
```

确定 Amazon API Gateway 访问日志组中运行时间最长的 10 个 Amazon API Gateway 请求

```
fields @timestamp, status, ip, path, httpMethod, responseLatency
| sort responseLatency desc
| limit 10
```

返回 Amazon API Gateway 访问日志组中最受欢迎的 API 路径列表

```
stats count(*) as requestCount by path
| sort requestCount desc
| limit 10
```

为 Amazon API Gateway 访问日志组创建集成延迟报告

```
filter status=200
| stats avg(integrationLatency), max(integrationLatency),
min(integrationLatency) by bin(1m)
```

NAT 网关的查询

如果您注意到您的 Amazon 账单高于正常成本，则可以使用 CloudWatch Logs Insights 查找主要耗用项。有关以下查询命令的更多信息，请参阅 Amazon 高级支持页面中的[如何在我的 VPC 中找到通过 NAT 网关的主要流量耗用项？](#)

Note

在以下查询命令中，将 "x.x.x.x" 替换为 NAT 网关的私有 IP，然后用 VPC CIDR 范围的前两个八位字节替换 "y.y"。

查找通过 NAT 网关发送流量最多的实例。

```
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
```

确定 NAT 网关中进出实例的流量。

```
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
```

确定 VPC 中实例最常与之进行上传和下载的互联网目标。

对于上载

```
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
```

对于下载

```
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
```

查询 Apache 服务器日志

您可以使用 CloudWatch Logs Insights 查询 Apache 服务器日志。有关以下查询的更多信息，请参阅 Amazon 云运营与迁移博客中的[使用 CloudWatch Logs Insights 简化 Apache 服务器日志](#)。

找到最相关的字段，以便您可以在应用程序的 /admin 路径中审阅访问日志并查看流量。

```
fields @timestamp, remoteIP, request, status, filename | sort @timestamp desc
| filter filename="/var/www/html/admin"
| limit 20
```

使用状态代码“200”（成功）查找访问主页的唯一 GET 请求数量。

```
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
```

查找 Apache 服务重新启动的次数。

```
fields @timestamp, function, process, message
| filter message like "resuming normal operations"
| sort @timestamp desc
| limit 20
```

针对 Amazon EventBridge 的查询

获取按事件详细信息类型分组的 EventBridge 事件的数量

```
fields @timestamp, @message
| stats count(*) as numberOfEvents by `detail-type`
| sort numberOfEvents desc
```

解析命令的示例

使用 glob 表达式来从日志字段 **@message** 提取字段 **@user**、**@method** 和 **@latency**，并对于 **@method** 和 **@user** 的每个唯一组合返回平均延迟。

```
parse @message "user=*, method:*, latency := *" as @user,
    @method, @latency | stats avg(@latency) by @method,
    @user
```

使用正则表达式从日志字段 **@message** 提取字段 **@user2**、**@method2** 和 **@latency2**，并对于 **@method2** 和 **@user2** 的每个唯一组合返回平均延迟。

```
parse @message /user=(?<user2>.*?), method:(?<method2>.*?),
    latency := (?<latency2>.*?)/ | stats avg(latency2) by @method2,
    @user2
```

提取字段 **loggingTime**、**loggingType** 和 **loggingMessage**，筛选出包含 **ERROR** 或 **INFO** 字符串的日志事件，然后仅显示包含 **ERROR** 字符串的事件的 **loggingMessage** 和 **loggingType** 字段。

```
FIELDS @message
| PARSE @message "*" [*] "*" as loggingTime, loggingType, loggingMessage
| FILTER loggingType IN ["ERROR", "INFO"]
| DISPLAY loggingMessage, loggingType = "ERROR" as isError
```

与之前的时间范围进行比较 (差异)

您可以将 CloudWatch Logs Insights 与 Logs Insights QL 结合使用来比较一段时间内日志事件的变化。您可以将最近时间范围内提取的日志事件与紧接的前一个时间段的日志进行比较。或者，您可以与类似的过去时间段进行比较。这可以帮助您发现日志中的错误是最近引入的还是已经发生的，并可以帮助您发现其他趋势。

比较查询仅返回结果中的模式，而不是原始日志事件。返回的模式将帮助您快速查看一段时间内日志事件的趋势和变化。运行比较查询并获得模式结果后，您可以查看自己感兴趣的模式的示例原始日志事件。有关日志模式的更多信息，请参阅 [模式分析](#)。

当您运行比较查询时，系统会根据两个不同的时间段分析您的查询：您选择的原始查询时间段和比较时间段。比较时间段的长度始终与原始查询时间段的长度相同。比较的默认时间间隔如下。

- 上一时间段 — 与查询时间段之前的时间段进行比较。
- 前一天 — 与查询时间段之前一天的时间段进行比较。
- 上一周 — 与查询时间段之前一周的时间段进行比较。
- 上个月 — 与查询时间段之前一个月的时间段进行比较。

Note

使用比较的查询产生的费用与在组合时间范围内运行单个 CloudWatch Logs Insights 查询的费用相似。有关更多信息，请参阅 [Amazon CloudWatch 定价](#)。

运行比较查询

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择日志、Logs Insights。

查询框中会出现默认查询。

3. 确认已选中 Logs Insights QL 选项卡。
4. 保留默认查询或输入其他查询。
5. 在选择日志组下拉菜单中，选择要查询的一个或多个日志组。
6. （可选）使用时间间隔选择器选择要查询的时间段。默认查询针对的是前一小时的日志数据。
7. 在时间范围选择器中，选择比较。然后选择要与原始日志进行比较的上一个时间段，并选择应用。
8. 选择运行查询。

为了使查询从比较时间段获取数据，将在您的查询中附加 diff 命令。

9. 选择模式选项卡以查看结果。

该表将显示以下信息：

- 每个模式，模式的可变部分替换为动态令牌符号 `<string-number>`。###是令牌所代表的数据类型描述。##显示了与其他动态令牌相比，此令牌在模式中的出现位置。有关更多信息，请参阅 [模式分析](#)。
- 事件计数是原始的、较新的时间段内具有该模式的日志事件的数量。

- 差异事件计数是当前时间段与比较时间段内匹配的日志事件数之间的差异。正差异表示当前时间段内此类事件较多。
- 差异描述简要总结了当前时间段和比较时间段之间的模式变化。
- 严重性类型是具有此模式的日志事件的可能严重性，基于在日志事件中发现的字词（例如 FATAL、ERROR 和 WARN）。

10. 要进一步检查列表中的某个模式，请在检查列中选择其中一个模式的图标。

将出现模式检查窗格并显示以下内容：

- 模式。选择模式内的一个令牌来分析该令牌的值。
- 直方图，显示查询的时间范围内模式的出现次数。这可以帮助您识别有趣的趋势，例如某种模式的出现突然增加。
- 日志示例选项卡显示一些与所选模式匹配的日志事件。
- 如果您已选择动态令牌，则令牌值选项卡会显示所选动态令牌的值。

Note

每个令牌最多可捕获 10 个令牌值。令牌计数可能不准确。CloudWatch Logs 使用概率计数器来生成令牌计数，而不是绝对值。

- 相关模式选项卡显示与您正在检查的模式几乎同时频繁出现的其他模式。例如，如果 ERROR 消息的模式通常伴随另一个标记为 INFO 并带有附加详细信息的日志事件，则此处显示该模式。

在图形中可视化日志数据

您可以使用条形图、折线图和堆叠面积图等可视化效果，更高效地确定日志数据中的模式。CloudWatch Logs Insights 为使用 stats 函数以及一个或多个聚合函数的查询生成可视化效果。有关更多信息，请参阅[统计数据](#)。

OpenSearch 管道处理语言 (PPL)

本节简要介绍如何使用 OpenSearch PPL 查询 CloudWatch Logs。借助 PPL，您可以使用管道连接的命令来检索、查询和分析数据，从而更轻松地理解和编写复杂的查询。其语法基于 Unix 管道，支持通过命令链实现数据的转换与处理。使用 PPL，您可以对数据进行筛选和聚合，并使用一组丰富的数学、字符串、日期、条件和其他函数进行分析。

OpenSearch PPL 只能用于查询标准日志类中的日志组。

Note

有关 CloudWatch Logs 中支持的所有 OpenSearch PPL 查询命令以及语法和限制的详细信息，请参阅《OpenSearch Service 开发人员指南》中的[支持的 PPL 命令](#)。

有关您可以使用的其他查询语言的信息，请参阅 [CloudWatch Logs Insights](#)、[OpenSearch Service SQL](#) 和 [CloudWatch Metrics Insights](#)

命令或函数	示例查询	描述
fields	<code>fields field1, field2</code>	显示一组需要投影的字段。
join	<code>LEFT JOIN left=l, right=r on l.id = r.id `join_right_lg` fields l.field_1, r.field_2</code>	将两个数据集联接在一起。
其中	<code>where field1="success" where field2 != "i-023fe0a90929d8822" fields field3, field4, field5,field6 head 1000</code>	根据指定的条件筛选数据。
stats	<code>stats count(), count(field1), min(field 1), max(field1), avg(field1) by field2 head 1000</code>	执行聚合和计算
parse	<code>parse field1 ".*/(?<field2>[^\r\n]+)\$" where field2 = "requestId" fields field1, field2 head 1000</code>	从字符串中提取正则表达式 (regex) 模式，并显示提取出的模式。提取出的模式可进一步用于创建新字段或筛选数据。
排序	<code>stats count(), count(field1), min(field 1) as field1Alias, max(`field1`), avg(`field1`) by field2 sort -field1Al ias head 1000</code>	按字段名称对显示的结果进行排序。使用 sort -

命令或函数	示例查询	描述
		FieldName 按降序排序。
eval	<code>eval field2 = field1 * 2 fields field1, field2 head 20</code>	修改或处理字段的值，并将其存储在不同的字段中。这有助于对列进行数学修改、对列应用字符串函数或对列应用日期函数。
rename	<code>rename field2 as field1 fields field1;</code>	重命名搜索结果中的一个或多个字段。
head	<code>fields `@message` head 20</code>	将显示的查询结果限制为前 N 行。
top	<code>top 2 field1 by field2</code>	查找字段中最频繁出现的值。
dedup	<code>dedup field1 fields field1, field2, field3</code>	根据指定的字段删除重复的条目。
rare	<code>rare field1 by field2</code>	查找字段列表中所有字段中出现频率最低的值。
subquery	<code>where field_1 IN [search source=`subquery_lg` fields field_2] fields id, field_1</code>	在 PPL 语句中执行复杂的嵌套查询。
trendline	<code>trendline sma(2, field1) as field1Alias</code>	计算字段的移动平均值。

命令或函数	示例查询	描述
eventStats	<code>eventstats sum(field1) by field2</code>	使用计算得出的汇总统计数据丰富事件数据。它会分析您事件中的指定字段，计算各种统计指标，然后将这些结果作为新字段附加到每个原始事件上。
扩展	<code>eval tags_array_string = json_extract('@message', '\$.tags') eval tags_array = json_array(json_extract(tags_string, '\$[0]'), json_extract(tags_string, '\$[1]')) expand tags_array as color_tags</code>	将包含多个值的字段拆分为单独的行，为指定字段中的每个值创建新行。
fillnull	<code>fields `@timestamp`, error_code, status_code fillnull using status_code = "UNKNOWN", error_code = "UNKNOWN"</code>	使用您提供的值填充 null 字段。可在一个或多个字段中使用。
flatten	<code>eval metadata_struct = json_object('size', json_extract(metadata_string, '\$.size'), 'color', json_extract(metadata_string, '\$.color')) flatten metadata_struct as (meta_size, meta_color)</code>	扁平化一个字段。该字段必须是以下类型： : struct<?, ?> 或 array<struct<?, ?>> 。
cidrmatch	<code>where cidrmatch(ip, '2003:db8::/32') fields ip</code>	检查指定的 IP 地址是否在给定的 CIDR 范围内。

命令或函数	示例查询	描述
fieldsummary	<pre>where field1 != 200 fieldsummary includefields= field1 nulls=true</pre>	计算每个字段的基本统计数据（计数、非重复计数、最小值、最大值、平均值、标准差和平均值）。
grok	<pre>grok email '.*@%{HOSTNAME:host}' fields email, host</pre>	使用 grok 模式解析文本字段，并将结果附加到搜索结果中。
字符串函数	<pre>eval field1Len = LENGTH(field1) fields field1Len</pre>	PPL 中的内置函数，可在 PPL 查询中操作和转换字符串及文本数据。例如，转换大小写、合并字符串、提取部分内容以及清理文本。

命令或函数	示例查询	描述
日期和时间函数	<code>eval newDate = ADDDATE(DATE('2020-08-26'), 1) fields newDate</code>	内置函数，用于处理和转换 PPL 查询中的日期和时间戳数据。例如，date_add、date_format、datediff、date_sub、timestampadd、timestampdiff、current_timezone、utc_timestamp 和 current_date。
条件函数	<code>eval field2 = isnull(field1) fields field2, field1, field3</code>	内置函数，用于检查特定字段条件并有条件地计算表达式。例如，如果 field1 为空值，则返回 field2。

命令或函数	示例查询	描述
数学函数	<code>eval field2 = ACOS(field1) fields field1</code>	内置函数，用于在 PPL 查询中执行数学计算和转换。例如：abs（绝对值）、round（四舍五入）、sqrt（平方根）、pow（乘方计算）和ceil（向上取整到最近的整数）。
CryptoGraphic 函数	<code>eval crypto = MD5(field) head 1000</code>	计算给定字段的哈希值
JSON 函数	<code>eval valid_json = json('[1,2,3,{"f1":1,"f2":[5,6]},4]') fields valid_json</code>	用于处理 JSON 的内置函数，包括数组、提取及验证。例如，json_object、json_array、to_json_string、json_array_length、json_extract、json_keys 和 json_valid。

OpenSearch 结构化查询语言 (SQL)

本节简要介绍如何使用 OpenSearch SQL 查询 CloudWatch Logs。如果您习惯使用关系数据库，则它可提供熟悉的选项。OpenSearch SQL 提供 SQL 功能的子集，使其成为执行临时查询和数据分析任务的理想选择。借助 OpenSearch SQL，您可以使用 SELECT、FROM、WHERE、GROUP BY、HAVING 等命令，以及各种其他 SQL 命令和函数。您可以跨日志组执行 JOIN 操作，使用子查询跨日志组关联数据，并使用丰富的 JSON、数学、字符串、条件和其他 SQL 函数集对日志和安全数据执行强大的分析。

OpenSearch SQL 只能用于查询标准日志类中的日志组。

 Note

下表列出了 CloudWatch Logs 中支持的 SQL 命令和函数。有关所有 OpenSearch SQL 命令（包括语法）的信息，请参阅《OpenSearch Service 开发人员指南》中的[支持的 SQL 命令](#)。有关您可以使用的其他查询语言的信息，请参阅 [CloudWatch Logs Insights](#)、[OpenSearch Service PPL](#) 和 [CloudWatch Metrics Insights](#)。

支持的 SQL 命令

 Note

在示例查询列中，根据需要替换 *<logGroup>*，具体取决于您要查询的数据来源。

命令或函数	示例查询	描述
SELECT	SELECT `@message`, Operation FROM `LogGroupA`	显示预测值。
FROM	SELECT `@message`, Operation FROM `LogGroupA`	内置子句，用于指定要从中检索数据的源表或视图，支持各种类型的联接和子查询。

命令或函数	示例查询	描述
WHERE	<pre>SELECT * FROM `LogGroupA` WHERE Operation = 'x'</pre>	根据提供的字段条件筛选日志事件。
GROUP BY	<pre>SELECT `@logStream`, COUNT(*) as log_count FROM `LogGroupA` GROUP BY `@logStream`</pre>	根据类别对日志事件进行分组，并根据统计数据计算平均值。
HAVING	<pre>SELECT `@logStream`, COUNT(*) as log_count FROM `LogGroupA` GROUP BY `@logStream` HAVING log_count > 100</pre>	根据分组条件筛选结果。
ORDER BY	<pre>SELECT * FROM `LogGroupA` ORDER BY `@timestamp` DESC</pre>	根据 ORDER BY 子句中的字段对结果进行排序。您可以按降序或升序进行排序。
JOIN	<pre>SELECT A.`@message`, B.`@timestamp` FROM `LogGroupA` as A INNER JOIN `LogGroupB` as B ON A.`requestId` = B.`requestId`</pre>	根据共同字段合并两个表的结果。必须指定内部联接或左外联接
LIMIT	<pre>Select * from `LogGroupA` limit 10</pre>	将显示的查询结果限制为前 N 行。

命令或函数	示例查询	描述
字符串函数	<pre>SELECT upper(Operation) , lower(Operation), Operation FROM `LogGroupA`</pre>	SQL 中的内置函数，可在 SQL 查询中操作和转换字符串及文本数据。例如，转换大小写、合并字符串、提取部分内容以及清理文本。
日期函数	<pre>SELECT current_date() as today, date_add(current_date(), 30) as thirty_days_later, last_day(current_date()) as month_end FROM `LogGroupA`</pre>	内置函数，用于处理和转换 SQL 查询中的日期和时间戳数据。例如，date_add、date_format、datediff 和 current_date。
条件函数	<pre>SELECT Operation, IF(Error > 0, 'High', 'Low') as error_category FROM `LogGroupA`;</pre>	内置函数，根据指定条件执行操作，或在特定条件下评估表达式。例如，CASE 和 IF。
聚合函数	<pre>SELECT AVG(bytes) as bytesWritten FROM `LogGroupA`</pre>	内置函数，可对多行数据执行计算以生成单个汇总值。例如，SUM、COUNT、AVG、MAX 和 MIN。

命令或函数	示例查询	描述
JSON 函数	<pre>SELECT get_json_object(json_column, '\$.name') as name FROM `LogGroupA`</pre>	内置函数，用于在 SQL 查询中解析、提取、修改和查询 JSON 格式的数据（例如，from_json、to_json、get_json_object、json_tuple），支持对数据集中的 JSON 结构进行操作。
数组函数	<pre>SELECT scores, size(scores) as length, array_contains(scores, 90) as has_90 FROM `LogGroupA`;</pre>	内置函数，用于在 SQL 查询中处理数组类型列，允许访问、修改和分析数组数据（例如，size、explode、array_contains）等操作。

命令或函数	示例查询	描述
窗口函数	<pre>SELECT field1, field2, RANK() OVER (ORDER BY field2 DESC) as field2Rank FROM `LogGroupA`;</pre>	内置函数，可在当前行（窗口）相关的指定行集上执行计算，支持排名、运行总数和移动平均值等操作。例如，ROW_NUMBER、RANK、LAG 和 LEAD
转换函数	<pre>SELECT CAST('123' AS INT) as converted_number, CAST(123 AS STRING) as converted_string FROM `LogGroupA`</pre>	内置函数，在 SQL 查询中用于将数据从一种类型转换为另一种类型，支持数据类型转换和格式转换。例如，CAST、TO_DATE、TO_TIMESTAMP 和 BINARY。
谓词函数	<pre>SELECT scores, size(scores) as length, array_contains(scores, 90) as has_90 FROM `LogGroupA`;</pre>	内置函数，用于计算条件，并根据指定的标准或模式返回布尔值 (true/false)。例如，IN、LIKE、BETWEEN、IS NULL 和 EXISTS。

命令或函数	示例查询	描述
选择多个日志组	<pre>SELECT lg1.field1, lg1.field2 from `logGroups( logGroupIdentifier: ['LogGroup1', 'LogGroup2'])` as lg1 where lg1.field3= "Success"</pre>	使您能够在 SELECT 语句中指定多个日志组

多日志组查询支持的 SQL

要支持在 SQL 中查询多个日志组的使用案例，可以使用 `logGroups` 命令。使用此语法，您可以通过在 FROM 命令中指定多个日志组来查询它们。

语法：

```
`logGroups(
  logGroupIdentifier: ['LogGroup1', 'LogGroup2', ... 'LogGroupn']
)
```

在此语法中，您可在 `logGroupIdentifier` 参数中指定最多 50 个日志组。要引用监控账户中的日志组，请使用 ARN 而非 LogGroup 名称。

示例查询：

```
SELECT LG1.Column1, LG1.Column2 from `logGroups(
  logGroupIdentifier: ['LogGroup1', 'LogGroup2']
)` as LG1 WHERE LG1.Column1 = 'ABC'
```

查询 CloudWatch Logs 时，不支持以下在 FROM 语句后使用涉及多个日志组的语法：

```
SELECT Column1, Column2 FROM 'LogGroup1', 'LogGroup2', ... 'LogGroupn'
WHERE Column1 = 'ABC'
```

限制

当您使用 OpenSearch SQL 在 CloudWatch Logs Insights 中进行查询时，以下限制适用。

- SELECT 语句中只能包含一个 JOIN。
- 仅支持一层嵌套子查询。

- 不支持用分号 (;) 分隔的多语句查询。
- 不支持包含仅大小写不同的相同字段名 (例如 field1 和 FIELD1) 的查询。

例如，不支持以下查询：

```
Select AWSAccountId, AwsAccountId from LogGroup
```

然而，由于两个日志组中的字段名 (@logStream) 完全相同，因此支持以下查询：

```
Select a.`@logStream`, b.`@logStream` from Table A INNER Join Table B on a.id = b.id
```

- 函数和表达式必须对字段名进行操作，并作为包含在 FROM 子句中指定日志组的 SELECT 语句的一部分。

例如，不支持此查询：

```
SELECT cos(10) FROM LogGroup
```

支持此查询：

```
SELECT cos(field1) FROM LogGroup
```

- 使用 SQL 或 PPL 命令时，需将特定字段用反引号括起来才能成功查询。包含特殊字符 (非字母和非数字) 的字段需要使用反引号。例如，对 @message、Operation.Export 和 Test::Field 使用反引号。纯字母名称的字段无需使用反引号。

包含简单字段的查询示例：

```
SELECT SessionToken, Operation, StartTime FROM `LogGroup-A`  
LIMIT 1000;
```

附加反引号的类似查询：

```
SELECT `@SessionToken`, `@Operation`, `@StartTime` FROM `LogGroup-A` LIMIT 1000;
```

使用自然语言生成和更新 CloudWatch Logs Insights 查询

CloudWatch Logs 支持自然语言查询功能，可帮助您生成和更新 [CloudWatch Logs Insights](#)、[OpenSearch Service PPL](#)、[OpenSearch Service SQL](#) 和 [CloudWatch Metrics Insights](#) 的查询。

通过此功能，您可以用通俗易懂的英语询问或描述要查找的 CloudWatch Logs 数据。自然语言功能可根据您输入的提示生成查询，并逐行说明查询的工作原理。您也可以更新查询以进一步调查数据。

您可以根据自己的环境输入提示，例如“按传输的字节数计算，前 100 个源 IP 地址是什么？”和“查找 10 个最慢的 Lambda 函数请求”。

Note

自然语言查询功能是一项区域性服务。对于某些区域，该功能会向美国的区域发起跨区域调用以处理查询提示。有关更多信息，请参阅 [Amazon CloudWatch 扩展了对自然语言查询结果汇总和查询生成的区域支持](#)。

要使用此功能生成 CloudWatch Logs Insights 查询，请打开 CloudWatch Logs Insights 查询编辑器，选择要查询的日志组，然后选择生成查询。

Important

要使用自然语言查询功能，您必须使用 [CloudWatchLogsFullAccess](#)、[CloudWatchLogsReadOnlyAccess](#)、[AdministratorAccess](#) 或 [ReadOnlyAccess](#) IAM 策略登录，或者拥有 `cloudwatch:GenerateQuery` 权限。

示例查询

本节中的示例描述了如何使用自然语言功能生成和更新查询。

Note

有关 CloudWatch Logs Insights 查询编辑器和语法的更多信息，请参阅 [CloudWatch Logs Insights 查询语法](#)。

示例：生成自然语言查询

要使用自然语言来生成查询，请输入提示并选择生成新查询。这些示例展示了执行基本搜索的查询。

提示

以下提示示例指示该功能搜索速度最慢的 10 个 Lambda 函数调用。

```
Find the 10 slowest requests
```

查询

下面是使用 CloudWatch Logs Insights 查询语言进行的查询，它是自然语言功能根据提示生成的。请注意在查询之前的注释中，提示是如何显示的。查询完成后，您可以阅读有关查询工作原理的说明。

```
# 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

要关闭提示和有关查询工作原理的说明显示，请使用编辑器中的齿轮图标。

提示

要生成 OpenSearch SQL 查询，请选择 OpenSearch SQL 选项卡，然后打开查询生成器提示框以输入自然语言提示。下面是一个使用自然语言功能生成 OpenSearch SQL 查询的提示示例。

```
Give me the number of errors and exceptions per hour
```

查询

下面是该提示生成的 SQL 查询，可用于查找每小时汇总的错误和异常数量：

```
SELECT DATE_FORMAT(`@timestamp`, 'yyyy-MM-dd HH') AS hour,
```

```
COUNT(*) AS error_count
FROM `/aws/lambda/CloudWatchOdysseyQueryGen`
WHERE `@message` LIKE '%error%'
      OR `@message` LIKE '%exception%'
GROUP BY DATE_FORMAT(`@timestamp`, 'yyyy-MM-dd HH')
ORDER BY hour
```

提示

要生成 OpenSearch PPL 查询，请选择 OpenSearch PPL 选项卡，然后打开查询生成器提示框以输入自然语言提示。下面是一个使用自然语言功能生成 OpenSearch PPL 查询的提示示例。

```
Give me all unique exception messages
```

查询

下面是该提示生成的 PPL 查询，您可以使用它来查找日志中的唯一异常消息：

```
dedup @message
| fields @message
```

示例：更新自然语言查询

您可以通过编辑初始提示，然后选择更新查询，来更新查询。

更新版提示

以下示例显示了先前提示的更新版本。此提示不指示搜索 10 个速度最慢的 Lambda 函数调用，而是指示该功能搜索 20 个速度最慢的 Lambda 函数调用，并包含另一列以包含其他日志事件。

```
Show top 20 slowest requests instead and display requestId as a column
```

更新版查询

下面是一个使用 CloudWatch Logs Insights 查询语言进行更新的查询的示例。请注意在更新版查询之前的注释中，更新版提示是如何显示的。查询完成后，您可以阅读有关原始查询是如何更新的说明。

```
# 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.
```

选择不使用您的数据来改善服务

您为训练 AI 模型和生成相关查询而提供的自然语言提示数据仅用于提供和维护您的服务。这些数据可用于提高 CloudWatch Logs Insights 的质量。您的信任、隐私和内容的安全性是我们最重视的问题。有关更多信息，请参阅 [Amazon Service Terms](#) 和 [Amazon responsible AI policy](#)。

通过创建 AI 服务选择退出策略，您可以选择不将您的内容用于开发自然语言查询或提高自然语言查询的质量。要选择退出对所有 CloudWatch Logs AI 功能（包括查询生成功能）进行的数据收集，您必须为 CloudWatch Logs 创建一个选择退出策略。有关更多信息，请参阅《Amazon Organizations 用户指南》中的 [AI 服务选择退出策略](#)。

支持的日志和发现的字段

CloudWatch Logs Insights 支持不同的日志类型。对于发送到 Amazon CloudWatch Logs 中标准类日志组的每个日志，CloudWatch Logs Insights 会自动生成五个系统字段：

- @message 包含原始未解析的日志事件。这等同于 [InputLogevent](#) 中的 message 字段。
- @timestamp 包含日志事件的 timestamp 字段中事件时间戳。这等同于 [InputLogevent](#) 中的 timestamp 字段。
- @ingestionTime 包含 CloudWatch Logs 接收日志事件的时间。
- @logStream 包含已将日志事件添加到的日志流的名称。日志流使用与生成日志的相同进程对日志进行分组。
- @log 是形式的日志组标识符。*account-id:log-group-name* 在查询多个日志组时，这可能对于确定特定事件属于哪个日志组非常有用。
- @entity 包含 [探索相关遥测](#) 功能的与实体相关的扁平化 JSON。

例如，此 JSON 可以代表一个实体。

```
{
  "Entity": {
```

```
"KeyAttributes": {
  "Type": "Service",
  "Name": "PetClinic"
},
"Attributes": {
  "PlatformType": "AWS::EC2",
  "EC2.InstanceId": "i-1234567890123"
}
}
```

对于此实体，提取的系统字段如下所示：

```
@entity.KeyAttributes.Type = Service
@entity.KeyAttributes.Name = PetClinic
@entity.Attributes.PlatformType = AWS::EC2
@entity.Attributes.EC2.InstanceId = i-1234567890123
```

Note

仅标准日志类中的日志组支持字段发现。有关日志类的更多信息，请参阅 [日志类](#)。

CloudWatch Logs Insights 在其生成的字段的开头插入 @ 符号。

对于许多日志类型，CloudWatch Logs 还会自动发现日志中包含的日志字段。这些自动发现字段如下表所示。

对于具有 CloudWatch Logs Insights 不会自动发现的字段的其他类型日志，您可以使用 `parse` 命令来提取字段和创建提取字段以用于该查询中。有关更多信息，请参阅 [CloudWatch Logs Insights 语法查询语法](#)。

如果发现的日志字段的名称以 @ 字符开头，则 CloudWatch Logs Insights 显示它时会在开头另外加上一个 @。例如，如果日志字段名称为 `@example.com`，则此字段名称显示为 `@@example.com`。

Note

除了 `@message`、`@timestamp` 或 `@log` 之外，您还可以为已发现的字段创建字段索引。有关字段索引的更多信息，请参阅 [创建字段索引以提高查询性能并减少扫描量](#)。

日志类型	发现的日志字段
Amazon VPC 流日志	@timestamp , @logStream , @message, accountId , endTime, interfaceId , logStatus , startTime , version, action, bytes, dstAddr, dstPort, packets, protocol, srcAddr, srcPort
Route 53 日志	@timestamp , @logStream , @message, edgeLocation , ednsClientSubnet , hostZoneId , protocol, queryName , queryTimestamp , queryType , resolverIp , responseCode , version
Lambda 日志	@timestamp , @logStream , @message, @requestId , @duration, @billedDuration , @type, @maxMemoryUsed , @memorySize 如果 Lambda 日志行包含 X-Ray 跟踪 ID，则表明它还包括以下字段：@xrayTraceId 和 @xraySegmentId。 CloudWatch Logs Insights 自动发现 Lambda 日志中的日志字段，但仅适用于每个日志事件中的第一个嵌入式 JSON 片段。如果 Lambda 日志事件包含多个 JSON 片段，您可以使用 parse 命令解析并提取日志字段。有关更多信息，请参阅 JSON 日志中的字段。
CloudTrail 日志 JSON 格式的日志	有关更多信息，请参阅 JSON 日志中的字段 。
其他日志类型	@timestamp , @ingestionTime , @logStream , @message, @log.

JSON 日志中的字段

借助 CloudWatch Logs Insights，您可以使用点表示法表示 JSON 字段。本节包含一个示例 JSON 事件和代码片段，向您展示如何使用点表示法访问 JSON 字段。

示例：JSON 事件

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

该示例 JSON 事件包含一个名为 `userIdentity` 的对象。`userIdentity` 包含一个名为 `type` 的字段。要使用点表示法表示 `type` 的值，请使用 `userIdentity.type`。

该示例 JSON 事件包含多个数组，它们展平到嵌套字段名称和值的列表。要表示 `requestParameters.instancesSet` 中第一个项目的 `instanceId` 值，请使用 `requestParameters.instancesSet.items.0.instanceId`。放置在字段 `instanceId` 之前的数字 `0` 是指字段 `items` 的值的索引位置。下面的示例包含一个代码片段，向您展示如何访问 JSON 日志事件中的嵌套 JSON 字段。

示例：查询

```
fields @timestamp, @message
| filter requestParameters.instancesSet.items.0.instanceId="i-abcde123"
| sort @timestamp desc
```

此代码片段显示的查询将点表示法与 `filter` 命令结合使用，以访问嵌套 JSON 字段 `instanceId` 的值。该查询将筛选 `instanceId` 的值等于 `"i-abcde123"` 的消息，并返回包含指定值的所有录入事件。

Note

CloudWatch Logs Insights 最多可以从 JSON 日志中提取 200 个日志事件字段。对于未提取的额外字段，您可以使用 `parse` 命令从消息字段中的原始未解析录入事件中提取这些字段。有关 `parse` 命令的更多信息，请参阅《Amazon CloudWatch 用户指南》中的[查询语法](#)。

创建字段索引以提高查询性能并减少扫描量

您可以为日志事件中的字段创建字段索引，以便高效地进行基于等式的搜索。然后，当您在 CloudWatch Logs Insights 查询中使用字段索引时，查询会尝试跳过处理已知不包含索引字段的日志事件。这样可以减少使用字段索引的查询的扫描量，从而更快地返回结果。这可以帮助您快速搜索数千个日志组中 PB 级的总日志，并更快地找到相关日志。适合编制索引的字段是您经常需要查询的字段。具有高基数值的字段也是字段索引的良好候选对象，因为使用这些字段索引的查询限制了与目标值匹配的日志事件，因此会更快完成。

例如，假设您为 `requestId` 创建了一个字段索引。然后，包含 `requestId = value` 或 `requestId IN [value, value, ...]` 的针对该日志组的任何 CloudWatch Logs Insights 查询将尝试仅处理已知包含该索引字段和查询值的日志事件，并且 CloudWatch Logs 过去已检测到该字段的值。

您还可以利用字段索引来创建对大量日志组的高效查询。当您在查询中使用 `filterIndex` 命令而不是 `filter` 命令时，查询将针对具有字段索引的日志事件的选定日志组运行。这些查询最多可以扫描

10,000 个日志组，您可以通过指定最多五个日志组名称前缀进行选择。如果这是 CloudWatch 跨账户可观测性中的监控账户，您可以选择所有源账户或指定单个源账户来选择日志组。

索引字段区分大小写。例如，字段索引 RequestId 将无法匹配包含 requestId 的日志事件。

仅 JSON 和服务日志的结构化日志格式支持字段索引。

CloudWatch Logs 会为标准日志类中的所有日志组提供默认字段索引。默认字段索引可自动用于以下字段：

- @logStream
- @aws.region
- @aws.account
- @source.log
- traceId

默认字段索引是对您在策略中定义的任何自定义字段索引的补充。默认字段索引不计入[字段索引配额](#)。

CloudWatch Logs 仅对创建索引策略后摄取的日志事件编制索引。它不会对创建策略之前摄取的日志事件编制索引。创建字段索引后，每个匹配的日志事件将从日志事件的摄取时间起保留编制索引 30 天。

Note

如果您在监控账户中创建了一个字段索引策略，则该策略不会用于关联源账户中的日志组。字段索引策略仅适用于创建该策略的账户。

本节的其余主题介绍了如何创建字段索引。有关如何在查询中引用字段索引的信息，请参阅[filterIndex](#)和[筛选](#)。

主题

- [字段索引语法和配额](#)
- [创建账户级字段索引策略](#)
- [创建日志组级字段索引策略](#)
- [创建查询时的日志组选择选项](#)

- [删除字段索引策略的影响](#)

字段索引语法和配额

您可以通过创建字段索引策略来创建字段索引。您可以创建适用于整个账户的账户级索引策略，也可以创建仅适用于单个日志组的策略。对于账户范围内的索引策略，您可以设置适用于账户中的所有日志组的索引策略。您还可以创建适用于账户中的部分日志组的账户级索引策略（按日志组名称的前缀选择）。如果在同一个账户中有多个账户级策略，则这些策略的日志组名称前缀不能重叠。

日志组级字段索引策略会覆盖账户级字段索引策略：如果创建了日志组级索引策略，则该日志组仅使用该策略，而忽略账户级策略。

日志事件与字段索引名称的匹配区分大小写。例如，字段索引 `RequestId` 将无法匹配包含 `requestId` 的日志事件。

您最多可以拥有 20 个账户级索引策略。如果您有多个账户级索引策略，并且它们按日志组名称前缀进行筛选，则其中任何两个策略都不能使用相同或重叠的日志组名称前缀。例如，如果您有一个策略筛选出以 `my-log` 开头的日志组，则不能再有一个字段索引策略筛选出以 `my-logpprod` 或 `my-logging` 开头的日志组。

如果您有一个没有名称前缀且适用于所有日志组的账户级索引策略，则无法创建其他账户级索引策略。

每个索引策略都有以下配额和限制：

- 策略中最多可以包含 20 个字段。
- 每个字段名称最多可以包含 100 个字符。
- 要在日志组中创建以 `@` 开头的自定义字段的索引，必须在字段名称的开头额外添加一个 `@` 来指定该字段。例如，如果您的日志事件包含名为 `@userId` 的字段，则必须指定 `@@userId` 才能为此字段创建索引。

生成的字段和保留字段

CloudWatch Logs Insights 会自动在每个日志事件中生成系统字段。这些生成的字段以 `@` 为前缀。有关生成的字段的更多信息，请参阅[支持的日志和发现的字段](#)。

在这些生成的字段中，以下字段支持用作字段索引：

- `@logStream`
- `@ingestionTime`

- @requestId
- @type
- @initDuration
- @duration
- @billedDuration
- @memorySize
- @maxMemoryUsed
- @xrayTraceId
- @xraySegmentId

要为这些生成的字段编制索引，指定它们时无需像对以 @ 开头的自定义字段那样添加额外的 @。例如，要为 @logStream 创建字段索引，只需将 @logStream 指定为字段索引即可。

CloudWatch Logs 会为标准日志类中的所有日志组提供默认字段索引。默认字段索引可自动用于以下字段：

- @logStream
- @aws.region
- @aws.account
- @source.log
- traceId

默认字段索引是对您在策略中定义的任何自定义字段索引的补充。默认字段索引不计入[字段索引配额](#)。

JSON 日志中的子字段和数组字段

您可以对 JSON 日志中的嵌套子字段或数组字段编制索引。

例如，您可以在此日志的 userIdentity 字段中创建 accessKeyId 子字段的索引：

```
{
  "eventVersion": "1.0",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
```

```

    "arn": "arn: aws: iam: : 123456789012: user/Alice",
    "accessKeyId": "11112222",
    "accountId": "123456789012",
    "userName": "Alice"
  },
  "eventTime": "2014-03-06T21: 22: 54Z",
  "eventSource": "ec2.amazonaws.com",
  "eventName": "StartInstances",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.255",
  "userAgent": "ec2-api-tools1.6.12.2",
  "requestParameters": {
    "instancesSet": {
      "items": [{
        "instanceId": "i-abcde123",
        "currentState": {
          "code": 0,
          "name": "pending"
        },
        "previousState": {
          "code": 80,
          "name": "stopped"
        }
      }]
    }
  }
}

```

要创建此字段，在创建字段索引和在查询中指定该字段时，请使用点表示法 (userIdentity.accessKeyId) 来引用它。查询可能如下所示：

```

fields @timestamp, @message
| filterIndex userIdentity.accessKeyId = "11112222"

```

在前面的示例事件中，instanceId 字段位于 requestParameters.instancesSet.items 中的一个数组中。为了在创建字段索引和查询时表示此字段，请将其引用为 requestParameters.instancesSet.items.0.instanceId。0 表示该字段在数组中的位置。

然后，此字段的查询可能如下所示：

```

fields @timestamp, @message
| filterIndex requestParameters.instancesSet.items.0.instanceId="i-abcde123"

```

创建账户级字段索引策略

请按照本节中的步骤创建适用于账户中的所有日志组或者日志组名称以相同字符串开头的多个日志组的字段索引策略。

创建账户级字段索引策略

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在左侧导航窗格上，选择设置，然后选择日志选项卡。
3. 在账户级索引策略部分，选择管理。
4. 选择创建索引策略。
5. 对于策略名称，输入新策略的名称。
6. 对于选择日志组，请执行下列操作之一：
 - 选择所有标准日志组，将索引策略应用于账户中的所有标准类日志组。
 - 选择按前缀匹配选择日志组，将策略应用于名称以相同字符串开头的日志组子集。然后，在输入前缀名称中输入这些日志组的前缀。

输入前缀后，您可以选择预览前缀匹配的日志组来确认您的前缀是否与预期日志组匹配。

7. 对于自定义索引字段配置，选择添加字段路径以输入要编制索引的第一个字段。

然后输入要用作字段名称值的字符串。这必须与日志事件中出现的內容的大小写完全匹配。例如，如果您的日志事件包含 `requestId`，则必须在此处输入 `requestId`。`RequestId`、`requestID` 和 `request Id` 不匹配。

如果要为以 `@` 字符开头的自定义日志字段编制索引，则在输入索引字符串时必须包含一个额外的 `@` 字符。例如，如果您有一个自定义日志字段 `@emailname`，请在添加字段路径框中输入 `@emailname`。

您还可以为 CloudWatch Logs 自动生成的 `@ingestionTime` 和 `@logStream` 字段创建索引。如果这样做，则在指定它们时无需添加额外的 `@`。

8. 重复上一步，最多可添加 20 个字段索引。
9. 完成后，选择 Create (创建)。

创建日志组级字段索引策略

请按照本节中的步骤创建适用于单个日志组的字段索引策略。

创建日志组级字段索引策略

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在左侧导航窗格中，依次选择 Logs (日志) 和 Log groups (日志组)。
3. 选择日志组的名称。
4. 选择字段索引选项卡。
5. 选择管理此日志组的字段索引
6. 对于管理日志组级字段索引，选择添加字段路径以输入要编制索引的第一个字段。

然后输入要用作字段名称值的字符串。这必须与日志事件中出现的內容的大小写完全匹配。例如，如果您的日志事件包含 `requestId`，则必须在此处输入 `requestId`。`RequestId`、`requestID` 和 `request Id` 匹配。

如果要为以 `@` 字符开头的自定义日志字段编制索引，则在输入索引字符串时必须包含一个额外的 `@` 字符。例如，如果您有一个自定义日志字段 `@emailname`，请在添加字段路径框中输入 `@emailname`。

您还可以为 CloudWatch Logs 自动生成的 `@ingestionTime` 和 `@logStream` 字段创建索引。如果这样做，则在指定它们时需要添加额外的 `@`。

7. 重复上一步，最多可添加 20 个字段索引。
8. 完成后，选择 Save。

创建查询时的日志组选择选项

本节介绍了选择要包含在查询中的日志组的各种方法。

在控制台中选择用于查询的日志组

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择日志、Logs Insights。
3. 选择要用于此查询的查询语言。您可以选择：Logs Insights QL、OpenSearch PPL 或 OpenSearch SQL。
4. 可通过三种方式选择要用于查询的日志组：
 - 使用日志组名称框。这是默认选择方法。使用此方法，您最多可以输入 50 个日志组名称。如果这是 CloudWatch 跨账户可观察性中的监控账户，则您可以在源账户和监控账户中选择日志组。单个查询可以同时查询来自不同账户的日志。

- 使用日志组条件部分。在此部分，您可以根据日志组名称的前缀选择日志组。一个查询最多可以包含五个前缀。名称中包含这些前缀的日志组将被选中。或者，所有日志组选项会选择账户中的所有日志组。
- 如果这是 CloudWatch 跨账户可观测性中的监控账户，您可以从账户下拉菜单中选择所有账户，以选择所有关联账户中的日志组。或者，您可以单独选择要包含在此查询中的账户。

如果您的选择与超过 10,000 个日志组匹配，您将看到一个错误，提示您缩小选择范围。

5. 查询的默认日志类为标准。您可以使用日志类将其更改为不频繁访问。

使用 Amazon CLI

要在从命令行启动查询时进行此类选择，可以在查询中使用 `source` 命令。有关更多信息以及示例，请参阅 [SOURCE](#)。

删除字段索引策略的影响

如果删除已生效一段时间的字段索引策略，则会发生以下情况：

- 在删除策略后的 30 天内，查询仍然可以从索引日志事件中受益。
- 如果您删除了日志组级索引策略，并且已存在适用于该日志组的账户级策略，则账户级策略最终将适用于该日志组。

模式分析

当您查询日志时，CloudWatch Logs Insights 会使用机器学习算法来查找模式。模式是在日志字段中重复出现的共有的文本结构。查看查询结果时，您可以选择模式选项卡来查看 CloudWatch Logs 根据结果示例找到的模式。或者，您可以将 `pattern` 命令附加到查询，以分析整组匹配日志事件中的模式。

模式对于分析大型日志集很有用，因为大量日志事件通常可压缩成几个模式。

考虑以下三个日志事件示例。

```
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
```

在上面的示例中，所有三个日志事件都遵循一种模式：

```
<Time-1> [INFO] Calling DynamoDB to store for resource id <ID-2>
```

模式内的字段称为令牌。模式内变化的字段（如请求 ID 或时间戳）是动态令牌。每个动态令牌都用 `<string-number>` 表示。`###` 是令牌所代表的数据类型的描述。`##` 显示了与其他动态令牌相比，此令牌在模式中的出现位置。

动态令牌的常见示例包括错误代码、时间戳以及请求 ID。令牌值代表动态令牌的特定值。例如，如果动态令牌表示 HTTP 错误代码，则令牌值可能是 501。

模式检测也用于 CloudWatch Logs 异常检测器和比较功能。有关更多信息，请参阅[日志异常检测](#)和[与之前的时间范围进行比较（差异）](#)。

开始使用模式分析

任何 CloudWatch Logs Insights 查询都会自动执行模式检测。不包含 `pattern` 命令的查询会同时在结果中获取日志事件和模式。

如果在查询中包含 `pattern` 命令，则会对整个匹配的日志事件集执行模式分析。这会为您提供更准确的模式结果，但是当您使用 `pattern` 命令时不会返回原始日志事件。当查询不包括 `pattern` 时，模式结果将基于返回的前 1000 个日志事件，或者基于您在查询中使用的限制值。如果查询中包括 `pattern`，则模式选项卡中显示的结果将来自查询匹配的所有日志事件。

开始使用 CloudWatch Logs Insights 中的模式分析

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择日志、Logs Insights。

在 Logs Insights（日志洞察）页面，查询编辑器包含一个默认查询，它将返回 20 个最近的日志事件。

3. 删除查询框中的 `| limit 20` 行，使得查询如下所示：

```
fields @timestamp, @message, @logStream, @log
| sort @timestamp desc
```

4. 在选择日志组下拉菜单中，选择要查询的一个或多个日志组。
5. （可选）使用时间间隔选择器选择要查询的时间段。

您可以选择 5 分钟到 30 分钟的间隔；1 小时、3 小时和 12 小时间隔；或者自定义时间范围。

6. 选择运行查询以启动查询。

查询运行完毕后，日志选项卡将显示查询返回的日志事件表。表上方会显示一条关于有多少条记录与查询匹配的消息，类似于显示匹配的 10,000 条记录，共 71,101 条。

7. 选择模式选项卡。

8. 该表现在会显示在查询中发现的模式。由于查询不包括 `pattern` 命令，因此此选项卡仅显示在日志选项卡的表中显示的 10,000 个日志事件中发现的模式。

对于每个模式，将显示下面的信息：

- 模式，每个动态令牌显示为 `<string-number>`。###是令牌所代表的数据类型的描述。##显示了与其他动态令牌相比，此令牌在模式中的出现位置。
- 事件计数，这是该模式在查询的日志事件中出现的次数。选择事件计数列标题可按频率对模式进行排序。
- 事件比率，即查询的日志事件中包含此模式的百分比。
- 严重性类型，将是以下项之一：
 - 如果模式包含错误一词，则为 ERROR。
 - 如果模式包含警告一词，但不包含错误，则为 WARN。
 - 如果模式不包含警告或错误，则为 INFO。

选择严重性信息列标题可按严重性对模式进行排序。

9. 现在更改查询。将查询中的 `| sort @timestamp desc` 行替换为 `| pattern @message`，以便完整查询如下：

```
fields @timestamp, @message, @logStream, @log
| pattern @message
```

10. 选择运行查询。

查询完成后，日志选项卡中没有结果。但是，模式选项卡可能会列出更多模式，具体取决于查询的日志事件总数。

11. 无论您是否在查询中包括 `pattern`，都可以进一步检查查询返回的模式。为此，请在检查列中选择其中一个模式的图标。

将出现模式检查窗格并显示以下内容：

- 模式。选择模式内的一个令牌来分析该令牌的值。

- 直方图，显示查询的时间范围内模式的出现次数。这可以帮助您识别有趣的趋势，例如某种模式的出现突然增加。
- 日志示例选项卡显示一些与所选模式匹配的日志事件。
- 如果您已选择动态令牌，则令牌值选项卡会显示所选动态令牌的值。

Note

每个令牌最多可捕获 10 个令牌值。令牌计数可能不准确。CloudWatch Logs 使用概率计数器来生成令牌计数，而不是绝对值。

- 相关模式选项卡显示与您正在检查的模式几乎同时频繁出现的其他模式。例如，如果 ERROR 消息的模式通常伴随另一个标记为 INFO 并带有附加详细信息的日志事件，则此处显示该模式。

有关 pattern 命令的详细信息

本节包含有关 pattern 命令及其用途的更多详细信息。

- 在上一个教程中，我们在添加 pattern 时删除了 sort 命令，因为如果查询在 sort 命令之后包含 pattern 命令，则查询无效。在 sort 排序之前包含 pattern 是有效的。

有关 pattern 语法的更多详细信息，请参阅 [模式](#)。

- 当在查询中使用 pattern 时，@message 必须是 pattern 命令中选择的字段之一。
- 可以在 pattern 命令之前包含 filter 命令，以便仅将筛选后的日志事件集用作模式分析的输入。
- 要查看特定字段（例如从 parse 命令派生的字段）的模式结果，请使用 pattern @fieldname。
- 具有非日志输出的查询（例如使用 stats 命令的查询）不会返回模式结果。

保存并重新运行 CloudWatch Logs Insights 查询

创建查询后，可以将其保存，以便稍后再次运行查询。查询将以文件夹结构保存，因此您可以组织它们。每个账户每个区域可以保存多达 1000 个查询。

查询保存在特定区域级别，而不是特定用户级别。如果您创建并保存了查询，则同一区域中有权访问 CloudWatch Logs 的其他用户可以查看该区域内所有已保存的查询及其文件夹结构。

要保存查询，您必须登录到具有 logs:PutQueryDefinition 权限的角色。要查看已保存查询的列表，您必须登录到具有 logs:DescribeQueryDefinitions 权限的角色。

保存查询

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Logs (日志)，然后选择 Logs Insights (日志洞察)。
3. 在查询编辑器中，创建查询。
4. 选择保存。

如果没有看到 Save (保存) 按钮，则需要更改为 CloudWatch Logs 控制台的新设计。为此，请执行以下操作：

- a. 在导航窗格中，选择 日志组。
 - b. 选择 Try the new design (试用新设计)。
 - c. 在导航窗格中，选择 Insights (洞察)，然后返回到此过程的第 3 步。
5. 输入查询的名称。
 6. (可选) 选择要保存查询的文件夹。选择 Create new (新建) 以创建文件夹。如果创建新文件夹，则可以在文件夹名称中使用斜杠 (/) 字符来定义文件夹结构。例如，命名新文件夹 **folder-level-1/folder-level-2** 会创建一个名为 **folder-level-1** 的顶级文件夹，并在该文件夹中创建另一个名为 **folder-level-2** 的文件夹。查询保存在 **folder-level-2** 中。
 7. (可选) 更改查询的日志组或查询文本。
 8. 选择保存。

Tip

您可以使用 PutQueryDefinition 为保存的查询创建文件夹。要为保存的查询创建文件夹，请使用正斜杠 (/) 在所需的查询名称前加上所需的文件夹名称：**<folder-name>/<query-name>**。有关此操作的更多信息，请参阅 [PutMetricAlarm](#)。

运行保存的查询

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Logs (日志)，然后选择 Logs Insights (日志洞察)。
3. 在右侧，选择 Queries (查询)。
4. 从 Saved queries (已保存的查询) 列表中选择您的查询。它显示在查询编辑器中。
5. 选择运行。

保存已保存查询的新版本

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Logs (日志)，然后选择 Logs Insights (日志洞察)。
3. 在右侧，选择 Queries (查询)。
4. 从 Saved queries (已保存的查询) 列表中选择您的查询。它显示在查询编辑器中。
5. 修改查询。如果您需要运行它来检查您的工作，请选择 Run query (运行查询)。
6. 准备好保存新版本后，选择 Actions (操作)、Save as (另存为)。
7. 输入查询的名称。
8. (可选) 选择要保存查询的文件夹。选择 Create new (新建) 以创建文件夹。如果创建新文件夹，则可以在文件夹名称中使用斜杠 (/) 字符来定义文件夹结构。例如，命名新文件夹 **folder-level-1/folder-level-2** 会创建一个名为 **folder-level-1** 的顶级文件夹，并在该文件夹中创建另一个名为 **folder-level-2** 的文件夹。查询保存在 **folder-level-2** 中。
9. (可选) 更改查询的日志组或查询文本。
10. 选择保存。

要删除查询，您必须登录到具有 logs:DeleteQueryDefinition 权限的角色。

编辑或删除已保存的查询

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Logs (日志)，然后选择 Logs Insights (日志洞察)。
3. 在右侧，选择 Queries (查询)。
4. 从 Saved queries (已保存的查询) 列表中选择您的查询。它显示在查询编辑器中。
5. 选择 Actions (操作)、Edit (编辑) 或 Actions (操作)、Delete (删除)。

将查询添加到控制面板或导出查询结果

在运行查询后，您可以将查询添加到 CloudWatch 控制面板，或者将结果复制到剪贴板。

添加到控制面板的查询会在您每次加载控制面板以及每次控制面板刷新时运行。这些查询计入 30 个并发 CloudWatch Logs Insights 查询的限制。

将查询结果添加到控制面板

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Logs (日志)，然后选择 Logs Insights (日志洞察)。
3. 选择一个或多个日志组并运行查询。
4. 选择 Add to dashboard (添加到控制面板)。
5. 选择控制面板，或者选择 Create new (新建) 为查询结果创建控制面板。
6. 选择要用于查询结果的小部件类型。
7. 输入小部件的名称。
8. 选择 Add to dashboard (添加到控制面板)。

将查询结果复制到剪贴板或者下载查询结果

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Logs (日志)，然后选择 Logs Insights (日志洞察)。
3. 选择一个或多个日志组并运行查询。
4. 选择 Export results (导出结果)，然后选择所需的选项。

查看正在运行的查询或查询历史记录

您可以查看当前正在进行的查询以及您最近的查询历史记录。

当前运行的查询包括您已添加到控制面板的查询。每个账户的并发 CloudWatch Logs Insights 查询限制为 30 个，包括添加到控制面板的查询在内。这 30 个查询中只有 15 个可以使用 OpenSearch Service PPL 或 OpenSearch Service SQL。

查看您的最近查询历史记录

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Logs (日志)，然后选择 Logs Insights (日志洞察)。
3. 如果您使用的是 CloudWatch Logs 控制台的新设计，请选择 History (历史记录)。如果您使用的是旧设计，请选择 Actions (操作)、View query history for this account (查看此账户的查询历史记录)。

此时将显示最近查询的列表。您可以通过选择查询并选择 Run (运行)，再次运行其中的任何一个查询。

在 Status (状态) 下，对于当前正在运行的所有查询，CloudWatch Logs 均显示 In progress (正在进行)。

使用 Amazon Key Management Service 加密查询结果

默认情况下，CloudWatch Logs 使用默认的 CloudWatch Logs 服务器端加密方法加密存储的 CloudWatch Logs Insights 查询结果。您可以选择改用 Amazon KMS 密钥来加密这些结果。如果您将 Amazon KMS 密钥与加密结果相关联，则 CloudWatch Logs 会使用该密钥对账户中所有查询的存储结果进行加密。

如果您稍后取消密钥与查询结果的关联，CloudWatch Logs 将对以后的查询再次使用默认加密方法。但是，在关联密钥时运行的查询仍使用该密钥加密。取消关联 KMS 密钥后，CloudWatch Logs 仍然可以返回这些结果，这是因为 CloudWatch Logs 仍然可以继续引用该密钥。但如果该密钥之后被禁用，则 CloudWatch Logs 无法读取使用该密钥加密的查询结果。

Important

CloudWatch Logs 仅支持对称 KMS 密钥。请勿使用非对称密钥来加密查询结果。有关更多信息，请参阅[使用对称和非对称密钥](#)。

限制

- 要执行下列步骤，您必须具有以下权限：`kms:CreateKey`、`kms:GetKeyPolicy` 和 `kms:PutKeyPolicy`。
- 将密钥与日志组关联或取消关联后，最多可能需要五分钟时间，此操作才能生效。
- 如果撤消 CloudWatch Logs 对关联密钥的访问权限或删除关联的 KMS 密钥，您将无法再检索 CloudWatch Logs 中的加密数据。
- 不能使用 CloudWatch 控制台关联密钥，必须使用 Amazon CLI 或 CloudWatch Logs API。

第 1 步：创建 Amazon KMS key

要创建 KMS，请使用以下 [create-key](#) 命令：

```
aws kms create-key
```

输出包含密钥的密钥 ID 和 Amazon 资源名称 (ARN)。下面是示例输出：

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

步骤 2：设置 KMS 密钥的权限

默认情况下，所有 KMS 密钥都是私有的。只有资源所有者可以使用它来加密和解密数据。但是，资源拥有者可以将密钥的访问权限授予其他用户和资源。在此步骤中，您将授予 CloudWatch Logs 服务主体权限以使用该密钥。此服务委托人必须位于存储密钥的相同 Amazon 区域内。

作为最佳做法，我们建议您将密钥的使用限制为仅限于您指定的 Amazon 账户。

首先，使用以下 `policy.json` 中的 `get-key-policy` [命令将您的 KMS 密钥默认策略保存为](#)：

```
aws kms get-key-policy --key-id key-id --policy-name default --output text > ./policy.json
```

在文本编辑器中打开 `policy.json` 文件，然后从以下语句之一中添加粗体的部分。使用逗号将现有语句与新语句分隔。这些语句使用 `Condition` 部分来增强 Amazon KMS 密钥的安全性。有关更多信息，请参阅 [Amazon KMS 密钥和加密上下文](#)。

此示例中的 `Condition` 部分限制了对指定账户 CloudWatch Logs Insights 查询结果的 Amazon KMS 密钥使用。

JSON

```
{
  "Version": "2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.region.amazonaws.com"
      },
      "Action": [
        "kms:Encrypt*",
        "kms:Decrypt*",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:Describe*"
      ],
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:logs:us-east-1:111122223333:query-
result:*"
        },
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

最后，使用以下 [put-key-policy](#) 命令添加更新后的策略：

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

步骤 3：将 KMS 密钥与查询结果关联

将 KMS 密钥与账户中的查询结果相关联

按如下方式使用 [disassociate-kms-key](#) 命令：

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

步骤 4：取消密钥与账户中查询结果的关联

要取消 KMS 密钥与查询结果的关联，请使用以下 [disassociate-kms-key](#) 命令：

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

根据 CloudWatch Logs Insights 查询结果生成自然语言摘要

分析日志数据对于了解应用程序的行为至关重要，但解释大量的日志条目可能很耗时。CloudWatch Logs Insights 现在提供一个自然语言摘要功能，可将复杂的查询结果转换为清晰简洁的摘要。此功能可帮助您快速识别问题，并从日志数据中获得切实可行的见解。

工作方式

CloudWatch Logs Insights 可以使用 Amazon Bedrock 根据查询结果生成用户可读的摘要。该功能支持所有 CloudWatch Logs Insights 查询语言，并根据您的日志数据提供清晰切实可行的见解。

区域可用性和数据处理

Important

使用此功能时，您的查询结果可能会在不同的 Amazon Web Services 区域进行处理。例如，如果您在美国东部（弗吉尼亚州北部）运行查询，则汇总信息可能会在美国西部（俄勒冈州）进行处理。

下表列出了查询结果功能可用的不同地理位置的可能处理 Amazon Web Services 区域：

支持的 CloudWatch Logs 地理位置	可能的处理区域
美国 (US)	US East (N. Virginia) Region 美国东部 (俄亥俄) 区域 US West (Oregon) Region
欧洲	欧洲地区 (法兰克福) 区域 欧洲地区 (爱尔兰) 区域 欧洲地区 (巴黎) 区域 欧洲地区 (斯德哥尔摩) 区域 欧洲地区 (伦敦) 区域
亚太地区	US East (N. Virginia) Region 美国东部 (俄亥俄) 区域 US West (Oregon) Region
南美洲	US East (N. Virginia) Region 美国东部 (俄亥俄) 区域 US West (Oregon) Region

入门

生成自然语言摘要

1. 运行您的 CloudWatch Logs Insights 查询。
2. 查询完成后，选择汇总结果。

权限

您必须拥有以下其中一项：

- CloudWatchLogsFullAccess 许可
- CloudWatchLogsReadOnlyAccess 许可
- 自定义 IAM 策略，包括
`cloudwatch:GenerateQueryResultsSummary`、`logs:GetQueryResults`、`logs:DescribeQueryResults` 和 `logs:FilterLogEvents` 操作

数据隐私

您的查询结果将得到安全处理，不会用于训练或改进 CloudWatch Logs Insights 或 Amazon Bedrock。如果您选择使用反馈按钮对查询结果摘要提供反馈，则您的反馈表明您对 CloudWatch Logs Insights 中提供的功能的满意度。

日志异常检测

您可以通过两种方式检测日志数据中的异常：创建日志异常检测器以进行持续监控，或者在 CloudWatch Logs Insights 查询中使用 [anomaly detection](#) 命令进行按需分析。

日志异常检测器会扫描摄取到日志组中的日志事件，并自动查找日志数据中的异常。异常检测使用机器学习和模式识别来建立典型日志内容的基准。对于按需分析，您可以使用 CloudWatch Logs Insights 查询中的 `anomaly detection` 命令来识别时间序列数据中的异常模式。有关基于查询的异常检测的更多信息，请参阅[在 CloudWatch Logs Insights 中使用异常检测](#)。

在为日志组创建异常检测器后，它使用日志组中过去两周的日志事件进行训练。训练时间可能长达 15 分钟。训练完成后，它会开始分析传入的日志以识别异常，并将异常显示在 CloudWatch Logs 控制台中供您检查。

CloudWatch Logs 模式识别通过识别日志中的静态和动态内容来提取日志模式。模式对于分析大型日志集很有用，因为大量日志事件通常可压缩成几个模式。

例如，请参阅以下三个日志事件示例。

```
2023-01-01 19:00:01 [INFO] Calling DynamoDB to store for ResourceID: 12342342k124-12345
2023-01-01 19:00:02 [INFO] Calling DynamoDB to store for ResourceID: 324892398123-1234R
2023-01-01 19:00:03 [INFO] Calling DynamoDB to store for ResourceID: 3ff231242342-12345
```

在上面的示例中，所有三个日志事件都遵循一种模式：

```
<Date-1> <Time-2> [INFO] Calling DynamoDB to store for resource id <ResourceID-3>
```

模式内的字段称为令牌。模式内变化的字段（如请求 ID 或时间戳）称为动态令牌。为动态令牌找到的每个不同值称为令牌值。

如果 CloudWatch Logs 可以推断动态令牌所代表的数据类型，它会将该令牌显示为 `<string-number>`。### 是令牌所代表的数据类型的描述。## 显示了与其他动态令牌相比，此令牌在模式中的出现位置。

CloudWatch Logs 根据分析包含该名称的日志事件的内容来分配名称的字符串部分。

如果 CloudWatch Logs 无法推断动态令牌所代表的数据类型，则会将令牌显示为 `<Token-##>`，## 表示此令牌与其他动态令牌相比在模式中出现的位置。

动态令牌的常见示例包括错误代码、IP 地址、时间戳以及请求 ID。

日志异常检测使用这些模式来查找异常。异常检测器模型训练期结束后，将根据已知趋势对日志进行评估。异常检测器将显著的波动标记为异常。

本章介绍了如何启用异常检测、查看异常、为日志异常检测器创建警报以及日志异常检测器发布的指标。它还介绍了如何使用 Amazon Key Management Service 对异常检测器及其结果进行加密。

创建日志异常检测器不会产生费用。

异常和模式的严重性和优先级

日志异常检测器发现的每个异常都分配有优先级。发现的每种模式都分配有严重性。

- 优先级是自动计算的，它基于模式的严重性级别以及与预期值的偏差量。例如，如果某个令牌值突然增加 500%，则即使异常的严重性为 NONE，也可能被指定为 HIGH 优先级。
- 严重性仅基于模式中找到的关键字，例如 FATAL、ERROR 和 WARN。如果未找到这些关键字，则模式的严重性将标记为 NONE。

异常可见性时间

创建异常检测器时，请为其指定最长异常可见性期限。这是异常在控制台中显示并由 [ListAnomalies](#) API 操作返回的天数。在异常经过此时间段后，如果异常继续发生，则会自动接受为正常行为，并且异常检测器模型会停止将其标记为异常。

如果在创建异常检测器时不调整可见性时间，则默认使用 21 天。

抑制异常

发现异常后，您可以选择暂时或永久抑制它。抑制异常会导致异常检测器在您指定的时间内停止将此事件标记为异常。抑制异常时，您可以选择仅抑制该特定异常，或抑制与发现异常的模式相关的所有异常。

您仍然可以在控制台中查看被抑制的异常。您也可以选择停止抑制它们。

常见问题

Amazon 是否会使用我的数据来训练机器学习算法以供 Amazon 或其他客户使用？

否。训练创建的异常检测模型基于日志组中的日志事件，并且仅在该日志组和该 Amazon 账户内使用。

哪些类型的日志事件适合进行异常检测？

日志异常检测适用于：应用程序日志和其他类型的日志，其中大多数日志条目符合典型模式。事件包含日志级别或严重性关键字（如 INFO、ERROR 和 DEBUG）的日志组尤其适合日志异常检测。

日志异常检测不适用于：具有极长 JSON 结构的日志事件，例如 CloudTrail 日志。模式分析仅分析日志行的前 1500 个字符，因此超出该限制的任何字符都将被跳过。

审计或访问日志（如 VPC 流日志）在异常检测方面的成功率也较低。异常检测旨在发现应用程序问题，因此它可能不太适合网络或访问异常。

为了帮助确定异常检测器是否适用于某个日志组，请使用 CloudWatch Logs 模式分析来查找该组中日志事件中的模式数量。如果模式的数量不超过 300 个，则异常检测可能会效果良好。有关模式分析的更多信息，请参阅 [模式分析](#)。

什么会被标记为异常？

以下情况可能会导致日志事件被标记为异常：

- 具有日志组中以前未见过的模式的日志事件。
- 已知模式的显著变化。
- 具有一组离散的常用值的动态令牌的新值。
- 动态令牌的值出现次数的大幅变化。

虽然上述所有项都可能被标记为异常，但它们并不一定意味着应用程序性能不佳。例如，高于平常的 200 个成功值可能会被标记为异常。在这种情况下，您可以考虑抑制这些不表示问题的异常。

被屏蔽的敏感数据会发生什么情况？

日志事件中被屏蔽为敏感数据的任何部分都不会进行异常扫描。有关屏蔽敏感数据的更多信息，请参阅 [通过屏蔽帮助保护敏感的日志数据](#)。

在 CloudWatch Logs Insights 中使用异常检测

除了创建日志异常检测器以进行持续监控外，您还可以使用 CloudWatch Logs Insights 查询中的 `anomaly` 命令按需识别日志数据中的异常模式。此命令扩展了现有 `pattern` 功能，并使用机器学习来检测五种类型的异常，包括模式频率变化、新模式和令牌变化。

`anomaly` 命令尤其适用于：

- 对历史日志数据进行临时分析，以识别异常模式

- 调查特定时间段内的异常行为
- 监控 Lambda 函数等应用程序的执行问题

有关在查询中使用 `anomaly` 命令的更多信息，请参阅 [异常](#)。

这种基于查询的异常检测是对以下各节中所述的连续异常检测器的补充，为您提供实时监控和按需分析功能。

对日志组启用异常检测

按照以下步骤使用 CloudWatch 控制台创建日志异常检测器，以扫描日志组是否存在异常。

您还可以以编程方式创建异常检测器。有关更多信息，请参阅 [createLogAnomalyDetector](#)。

创建日志异常检测器

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 选择日志、日志异常。
3. 选择创建异常检测器。
4. 选择要为其创建此异常检测器的日志组。
5. 在异常检测器名称中输入检测器的名称。
6. （可选）将评估频率从默认 5 分钟更改为其他频率。根据日志组接收新日志的频率设置此值。例如，如果日志组每 10 分钟批量接收新的日志事件，则将评估频率设置为 15 分钟可能合适。
7. （可选）要将异常检测器配置为仅在包含特定单词或字符串的日志事件中查找异常，请选择筛选模式。

然后，在异常检测筛选模式中输入模式。有关模式语法的更多信息，请参阅 [指标筛选条件、订阅筛选条件、筛选日志事件和 Live Tail 的筛选条件模式语法](#)。

（可选）要测试筛选模式，请在日志事件消息中输入一些日志消息，然后选择测试模式。

8. （可选）要更改默认异常可见性期限或将 Amazon KMS 密钥与此异常检测器关联，请选择高级配置。
 - a. 要更改默认异常可见性期限，请在最长异常可见性期限（天）中输入一个新值。
 - b. 要将 Amazon KMS 密钥与此异常检测器关联，请在 KMS 密钥 ARN 中输入 ARN。如果您分配了一个密钥，则此检测器发现的异常信息将使用该密钥进行静态加密。用户必须拥有此密钥的权限以及异常检测器检索其发现的异常相关信息的权限。

您还必须确保 CloudWatch Logs 服务主体有权使用该密钥。有关更多信息，请参阅 [使用 Amazon KMS 对异常检测器及其结果进行加密](#)。

9. 选择启用异常检测。

根据日志组正在提取的日志事件，将创建异常检测器并开始训练其模型。大约 15 分钟后，异常检测开始启动并开始查找和发现异常。

查看已发现的异常

创建一个或多个日志异常检测器后，您可以使用 CloudWatch 控制台查看它们发现的异常。

您可以通过编程方式查看异常。有关更多信息，请参阅 [ListAnomalies](#)。

查看所有日志异常检测器发现的异常

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 选择日志、日志异常。

将出现日志异常表。日志异常旁边顶部的数字显示表中列出了多少个日志异常。表中的每一行显示以下信息：

- 异常列显示异常的简短摘要。这些摘要由 CloudWatch Logs 生成。
- 异常的优先级。优先级是根据日志事件的变化量、关键字（例如日志事件中发生的 Exception）等自动计算的。
- 异常基于的日志模式。有关模式的更多信息，请参阅 [日志异常检测](#)。
- 异常日志趋势显示一个直方图，描述与模式匹配的日志量。
- 上次检测时间显示最近一次发现此异常的时间。
- 首次检测时间显示首次发现此异常的时间。
- 异常检测器显示包含与此异常相关的日志事件的日志组的名称。您可以选择此名称来查看日志组详细信息页面。

3. 要进一步检查一个异常，请选择其行中的单选按钮。

将出现模式检查窗格并显示以下内容：

- 此异常基于的模式。选择模式内的一个令牌来分析该令牌的值。
- 直方图，显示查询的时间范围内异常的出现次数。

- 日志示例选项卡显示一些属于异常一部分的日志事件。
- 如果您已选择动态令牌，则令牌值选项卡会显示所选动态令牌的值。

 Note

每个令牌最多可捕获 10 个令牌值。令牌计数可能不准确。CloudWatch Logs 使用概率计数器来生成令牌计数，而不是绝对值。

4. 要抑制异常，请选择其行中的单选按钮，然后执行以下操作：
 - a. 选择操作、抑制异常。
 - b. 然后指定希望抑制异常的时长。
 - c. 要隐藏与此模式相关的所有异常，请选择抑制模式。
 - d. 选择抑制异常。

查看单个日志组中发现的异常

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 依次选择日志、日志组。
3. 选择日志组的名称，然后选择异常检测选项卡。

将出现异常检测表。日志异常旁边顶部的数字显示表中列出了多少个日志异常。表中的每一行显示以下信息：

- 异常列显示异常的简短摘要。这些摘要由 CloudWatch Logs 生成。
 - 异常的优先级。优先级是根据日志事件的变化量、关键字（例如日志事件中发生的 Exception）等自动计算的。
 - 异常基于的日志模式。有关模式的更多信息，请参阅 [日志异常检测](#)。
 - 异常日志趋势显示一个直方图，描述与模式匹配的日志量。
 - 上次检测时间显示最近一次发现此异常的时间。
 - 首次检测时间显示首次发现此异常的时间。
4. 要进一步检查一个异常，请选择其行中的单选按钮。

将出现模式检查窗格并显示以下内容：

- 此异常基于的模式。选择模式内的一个令牌来分析该令牌的值。

- 直方图，显示查询的时间范围内异常的出现次数。
- 日志示例选项卡显示一些属于异常一部分的日志事件。
- 如果您已选择动态令牌，则令牌值选项卡会显示所选动态令牌的值。

 Note

每个令牌最多可捕获 10 个令牌值。令牌计数可能不准确。CloudWatch Logs 使用概率计数器来生成令牌计数，而不是绝对值。

5. 要抑制异常，请选择其行中的单选按钮，然后执行以下操作：
 - a. 选择操作、抑制异常。
 - b. 然后指定希望抑制异常的时长。
 - c. 要隐藏与此模式相关的所有异常，请选择抑制模式。
 - d. 选择抑制异常。

在日志异常检测器上创建警报

您可以为日志组中的日志异常检测器创建警报。您可以指定当指定时间段内在日志组中发现指定数量的异常时，警报进入 ALARM 状态。您还可以使用筛选条件，以便警报仅计算指定优先级的异常。

为日志异常检测器创建警报

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择日志、日志异常。

将显示日志异常检测器表。

3. 选中要设置警报的异常检测器的单选按钮，然后选择创建警报。

将出现 CloudWatch 警报创建向导。LogAnomalyDetector 字段显示您选择的异常检测器的名称。指标名称字段显示 AnomalyCount。

4. （可选）要按异常优先级筛选此警报，请执行以下操作之一：
 - 要使警报仅计算高优先级异常，请对 LogAnomalyPriority 输入 **HIGH**。
 - 要使警报仅计算高优先级和中优先级异常，请对 LogAnomalyPriority 输入 **MEDIUM**。

有关优先级级别的更多信息，请参阅 [异常和模式的严重性和优先级](#)。

5. 选择对警报使用静态或指标异常检测阈值。此选择决定如何设置警报阈值。静态阈值意味着警报阈值是您选择的静态恒定数字。异常检测阈值意味着 CloudWatch 确定一个正常值范围，如果实际计数超过该范围的阈值，则会触发警报。您不必为日志异常检测警报选择异常检测。有关指标异常检测的更多信息，请参阅[使用 CloudWatch 异常检测](#)。
6. 对于每当 ***your-metric-name*** 为.....，选择大于、大于/等于、小于/等于或小于。然后，对于相比.....，为您的阈值指定一个数值。如果异常检测器在周期指定的时间内发现超过此数量的警报，则警报进入 ALARM 状态。
7. 选择其他配置。对于触发警报的数据点数，指定必须有多少个评估期（数据点）处于 ALARM 状态才能触发警报。如果此处的两个值匹配，则会创建一个告警；如果多个连续评估期违例，该告警将变为 ALARM（告警）状态。

要创建“M（最大为 N）”告警，为第一个值指定的数字应小于为第二个值指定的数字。有关更多信息，请参阅[评估告警](#)。

8. 对于 Missing data treatment（缺失数据处理），选择在缺失某些数据点时的告警行为。有关更多信息，请参阅[配置 CloudWatch 告警处理缺失数据的方式](#)。
9. 选择下一步。
10. 对于通知，选择添加通知，然后指定当警报转换为 ALARM、OK 或 INSUFFICIENT_DATA 状态时要通知的 Amazon SNS 主题。
 - a. （可选）要为相同告警状态或不同告警状态发送多个通知，请选择 Add notification（添加通知）。

 Note

我们建议您将警报设置为在进入数据不足状态以及进入警报状态时采取行动。这是因为连接到数据源的 Lambda 函数的许多问题都可能导致警报转换为数据不足。

- b. （可选）如果无需发送 Amazon SNS 通知，请选择移除。
11. （可选）如果您希望告警对 Amazon EC2 Auto Scaling、Amazon EC2、票证或 Amazon Systems Manager 执行操作，请选择相应的按钮，并指定告警状态和操作。

 Note

您的告警只有在处于 ALARM 状态时才能执行 Systems Manager 操作。有关 Systems Manager 操作的信息，请参阅[将 CloudWatch 配置为创建 OpsItems](#) 和[事件创建](#)。

12. 选择下一步。

13. 在名称和描述下，输入警报的名称和描述，然后选择下一步。名称必须仅包含 UTF-8 字符，并且不能包含 ASCII 控制字符。描述可以包含 Markdown 格式，该格式仅在 CloudWatch 控制台的警报详细信息选项卡中显示。Markdown 非常适合用于向运行手册或其他内部资源添加链接。

 Tip

警报名称只能包含 UTF-8 字符。不能包含 ASCII 控制字符。

14. 在预览和创建下，确认警报的信息和条件正确，然后选择创建警报。

日志异常检测器发布的指标

CloudWatch Logs 将 AnomalyCount 指标发布到 CloudWatch 指标。此指标会发布到 AWS/Logs 命名空间。

AnomalyCount 指标发布时包含以下维度：

- LogAnomalyDetector – 异常探测器的名称
- LogAnomalyPriority – 异常的优先级

使用 Amazon KMS 对异常检测器及其结果进行加密

异常检测器数据在 CloudWatch Logs 中始终是加密的。默认情况下，CloudWatch Logs 对静态数据使用服务器端加密。此外，您也可以使用 Amazon Key Management Service 进行这一加密。如果这样做，则使用 Amazon KMS 密钥进行加密。通过将 KMS 密钥与异常检测器关联，可以在异常检测器级别启用使用 Amazon KMS 进行加密。

 Important

CloudWatch Logs 仅支持对称 KMS 密钥。请勿使用非对称密钥对日志组中的数据进行加密。有关更多信息，请参阅[使用对称和非对称密钥](#)。

限制

- 要执行下列步骤，您必须具有以下权限：`kms:CreateKey`、`kms:GetKeyPolicy` 和 `kms:PutKeyPolicy`。

- 将密钥与异常检测器关联或取消关联后，该操作最多可能需要五分钟才能生效。
- 如果撤消 CloudWatch Logs 对关联密钥的访问权限或删除关联的 KMS 密钥，您将无法再检索 CloudWatch Logs 中的加密数据。

步骤 1：创建 Amazon KMS 密钥

要创建 KMS 密钥，请使用以下 [create-key](#) 命令：

```
aws kms create-key
```

输出包含密钥的密钥 ID 和 Amazon 资源名称 (ARN)。下面是示例输出：

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

步骤 2：设置 KMS 密钥的权限

默认情况下，所有 Amazon KMS 密钥都是私有的。只有资源所有者可以使用它来加密和解密数据。但是，资源拥有者可以将 KMS 密钥的访问权限授予其他用户和资源。在此步骤中，您将授予 CloudWatch Logs 服务主体权限以使用该密钥。此服务主体必须位于存储 KMS 密钥的相同 Amazon 区域内。

作为最佳实践，我们建议您将 KMS 密钥的使用限制为您指定的 Amazon 账户或异常检测器。

首先，使用以下 `policy.json` [get-key-policy](#) 命令将您的 KMS 密钥默认策略保存为：


```
aws kms get-key-policy --key-id key-id --policy-name default --output text > ./policy.json
```

在文本编辑器中打开 policy.json 文件，然后从以下语句之一中添加粗体的部分。使用逗号将现有语句与新语句分隔。这些语句使用 Condition 部分来增强 Amazon KMS 密钥的安全性。有关更多信息，请参阅 [Amazon KMS 密钥和加密上下文](#)。

此示例中的 Condition 部分将 Amazon KMS 密钥的使用限制于指定账户，但它可用于任何异常检测器。

JSON

```
{
  "Version": "2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Sid": "EnableIAMUserPermissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "AllowCloudWatchLogsEncryption",
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      },
    }
  ]
}
```

```

        "StringLike": {
            "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:123456789012:anomaly-detector:*"
        },
        "ArnLike": {
            "aws:SourceArn": "arn:aws:logs:us-east-1:123456789012:anomaly-detector:*"
        }
    },
    {
        "Sid": "AllowCloudWatchLogsDescribeKey",
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.us-east-1.amazonaws.com"
        },
        "Action": "kms:DescribeKey",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "123456789012"
            }
        }
    },
    {
        "Sid": "AllowCloudWatchLogsReEncryption",
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.us-east-1.amazonaws.com"
        },
        "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "123456789012"
            },
            "StringLike": {
                "kms:EncryptionContext:aws-crypto-ec:aws:logs:arn":
                    "arn:aws:logs:us-east-1:123456789012:anomaly-detector:*"
            }
        }
    }
}

```

```

        },
        "ArnLike": {
            "aws:SourceArn": "arn:aws:logs:us-east-1:123456789012:anomaly-detector:*"
        }
    },
    {
        "Sid": "AllowCloudWatchLogsDescribeKeyForReEncryption",
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.us-east-1.amazonaws.com"
        },
        "Action": "kms:DescribeKey",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "123456789012"
            }
        }
    }
]
}

```

最后，使用以下 [put-key-policy](#) 命令添加更新后的策略：

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

步骤 3：将 KMS 密钥与异常检测器关联

当在控制台中或者使用 Amazon CLI 或 API 创建 KMS 密钥时，您可以将 KMS 密钥与异常检测器关联。

步骤 4：取消密钥与异常检测器的关联

当密钥与异常检测器关联后，您无法更新该密钥。移除密钥的唯一方法是删除异常检测器，然后重新创建它。

使用 CloudWatch Logs Live Tail 进行故障排除

CloudWatch Logs Live Tail 通过查看提取的新日志事件的流式列表来帮助您快速处理事件。您可以近乎实时地查看、筛选和突出显示提取的日志，帮助您快速检测并解决问题。您可以根据指定的术语筛选日志，也可以突出显示包含指定术语的日志，以帮助快速找到所需内容。

Live Tail 会话按会话使用时间每分钟产生费用。有关定价的更多信息，请参阅 [Amazon CloudWatch 定价](#) 中的日志选项卡。

仅标准日志类中的日志组支持 Live Tail。有关日志类的更多信息，请参阅 [日志类](#)。

以下部分介绍了如何在控制台和 Amazon CLI 中使用 Live Tail。您还可以通过编程方式启动 Live Tail 会话。有关更多信息，请参阅 [StartLiveTail](#)。有关 SDK 示例，请参阅 [使用 Amazon SDK 启动 Live Tail 会话](#)。

您也可以在 Amazon Toolkit for Visual Studio Code 中使用 Live Tail。要从 VS Code 命令面板启动 Live Tail 会话，请参阅《Amazon Toolkit for Visual Studio Code 用户指南》的 [Amazon CloudWatch Logs Live Tail 部分](#)。

Live Tail 功能适用于所有商业 Amazon [区域](#)。它在中国区域或 Amazon GovCloud (美国) 区域中不可用。

Note

StartLiveTail API 使用 SDK 主机前缀注入将请求路由到 streaming-logs.*Region*.amazonaws.com。此 API 不支持 VPC 端点。

使用 Amazon CLI SDK 启动 Live Tail 会话

start-live-tail Amazon CLI 命令会为终端中的一个或多个日志组启动 Live Tail 流式传输会话。Live Tail 会话最长可持续三个小时。如果每秒有超过 500 个日志事件与筛选条件匹配，则显示的日志事件是总日志事件的样本，以提供实时的跟踪体验。有关 start-live-tail 命令的更多信息，请参阅 [start-live-tail](#)

您可以在两种模式下使用 start-live-tail：

- print-only – 这是默认模式

- interactive

print-only

在 print-only 模式下，日志事件在终端上流式传输。每秒都会在底部添加新事件，从而创建类似于 Linux 上的 `tail -f` 的近乎实时的跟踪体验。

要在 print-only 模式下启动 Live Tail 会话，请输入以下命令。

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-east-1:111111222222:log-group:my-logs
```

当使用 print-only 模式时，您还可以将其与其他 Linux 命令一起传输以增强其分析能力。以下示例使用 `error` 关键字筛选日志事件，并打印这些事件的第二列和第四列，以帮助提取特定信息。

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-east-1:111111222222:log-group:my-logs --mode print-only | grep "error" | awk '{print $2, $4}'
```

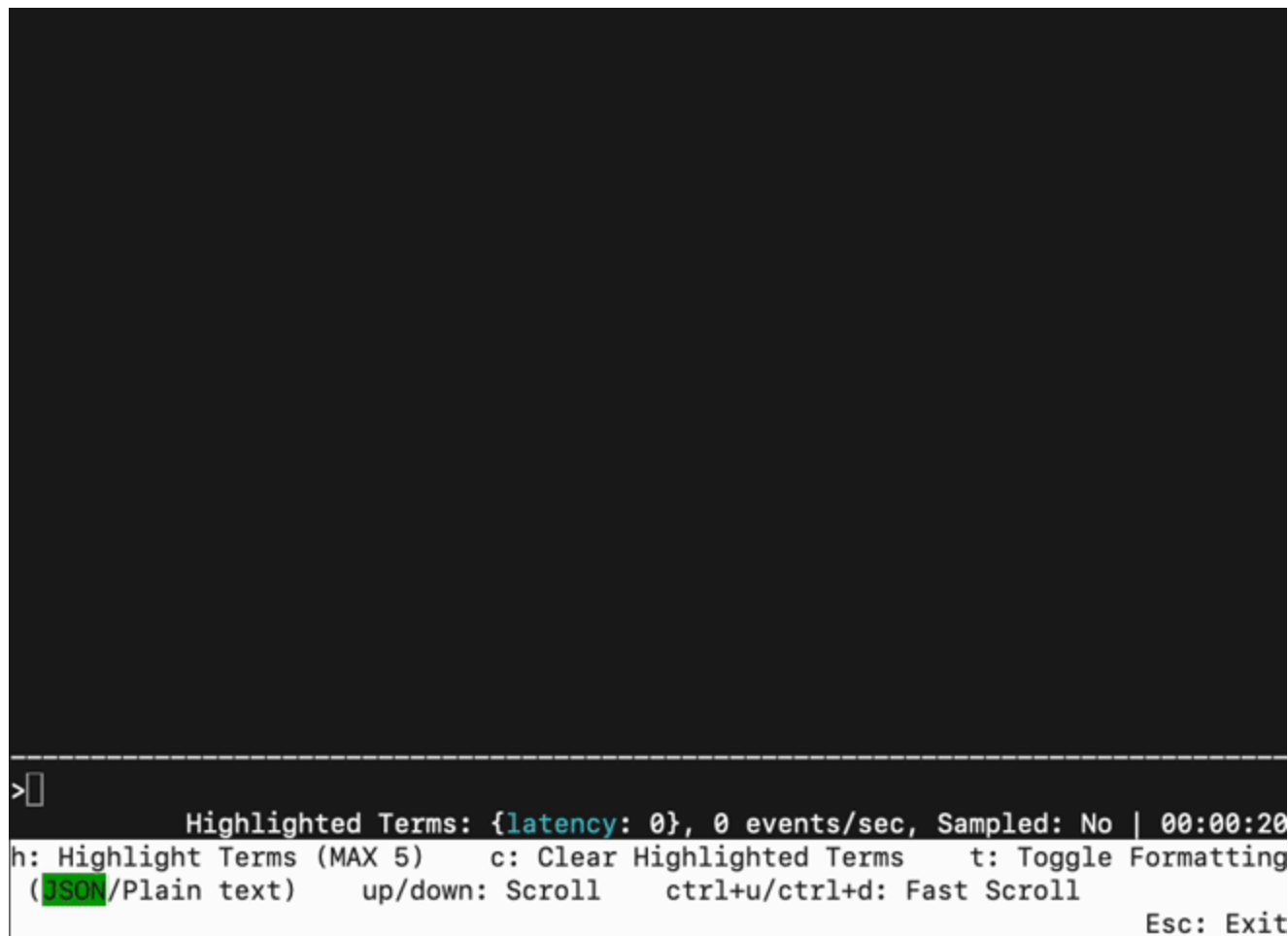
交互

在 interactive 模式下，您可以突出显示术语并在 JSON 和纯文本之间切换输出日志事件的格式。Interactive 模式还显示有关 Live Tail 会话的信息，例如会话持续时间、是否正在对会话进行采样、当前突出显示的术语和遇到这些术语的次数。

要在 interactive 模式下启动 Live Tail 会话，请输入以下命令。

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-east-1:111111222222:log-group:my-logs --mode interactive
```

Live Tail 会话开始。以下视频展示了示例会话的一部分。



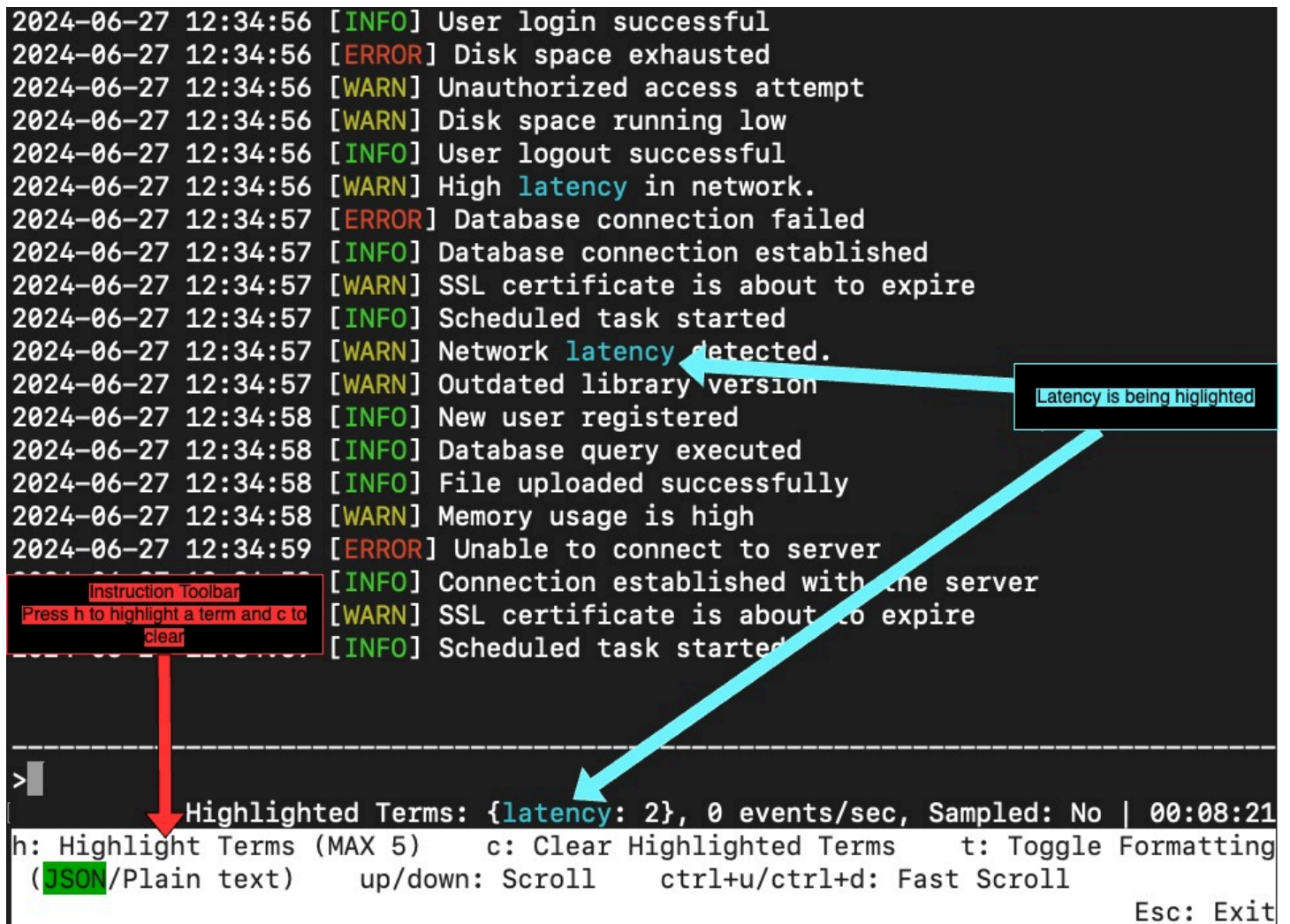
要突出显示流日志中的某个术语，请按 **h**，然后输入该术语。下面显示了突出显示术语 **latency** 后的屏幕。

要清除突出显示的术语，请按 **c**，然后键入代表要停止突出显示的术语的数字。

您可以按 **t** 在 JSON 和纯文本之间切换传入事件的显示格式。此切换功能是尽力而为的，并且仅当日志事件格式兼容时才会发生。

您可以使用向上和向下箭头键滚动，并使用 **CTRL+u** 和 **CTRL+d** 更快地滚动。

下图显示了 Live Tail 会话期间 **latency** 术语的突出显示。



```
2024-06-27 12:34:56 [INFO] User login successful
2024-06-27 12:34:56 [ERROR] Disk space exhausted
2024-06-27 12:34:56 [WARN] Unauthorized access attempt
2024-06-27 12:34:56 [WARN] Disk space running low
2024-06-27 12:34:56 [INFO] User logout successful
2024-06-27 12:34:56 [WARN] High latency in network.
2024-06-27 12:34:57 [ERROR] Database connection failed
2024-06-27 12:34:57 [INFO] Database connection established
2024-06-27 12:34:57 [WARN] SSL certificate is about to expire
2024-06-27 12:34:57 [INFO] Scheduled task started
2024-06-27 12:34:57 [WARN] Network latency detected.
2024-06-27 12:34:57 [WARN] Outdated library version
2024-06-27 12:34:58 [INFO] New user registered
2024-06-27 12:34:58 [INFO] Database query executed
2024-06-27 12:34:58 [INFO] File uploaded successfully
2024-06-27 12:34:58 [WARN] Memory usage is high
2024-06-27 12:34:59 [ERROR] Unable to connect to server
[INFO] Connection established with the server
[WARN] SSL certificate is about to expire
[INFO] Scheduled task started
```

Instruction Toolbar
Press h to highlight a term and c to clear

Highlighted Terms: {latency: 2}, 0 events/sec, Sampled: No | 00:08:21

h: Highlight Terms (MAX 5) c: Clear Highlighted Terms t: Toggle Formatting
(JSON/Plain text) up/down: Scroll ctrl+u/ctrl+d: Fast Scroll Esc: Exit

Latency is being highlighted

在控制台中启动 Live Tail 会话

您可以使用 CloudWatch 控制台启动 Live Tail 会话。以下程序说明了如何使用左侧导航窗格中的 Live tail 选项启动 Live Tail 会话。您也可以从“日志组”页面或 CloudWatch Logs Insights 页面启动 Live Tail 会话。

如果您使用数据保护策略来屏蔽使用 Live Tail 查看的日志组中的敏感数据，则敏感数据在 Live Tail 会话中始终会被屏蔽。有关屏蔽日志组中敏感数据的详细信息，请参阅 [通过屏蔽帮助保护敏感的日志数据](#)。

⚠ Important

如果您的网络安全团队不允许使用 Web 套接字，则您目前无法访问 CloudWatch 控制台的 Live Tail 部分。您可以将 Live Tail 与 Amazon CLI 或 API 一起使用。有关更多信息，请参阅 [使用 Amazon CLI SDK 启动 Live Tail 会话](#) 和 [StartLiveTail](#)。

开始 Live Tail 会话

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择日志、Live tail。
3. 对于选择日志组，在 Live Tail 会话中选择要从中查看事件的日志组。您可以选择多达 10 个日志组。
4. （可选）如果您只选择了一个日志组，则可以通过选择一个或多个日志流来查看日志事件，从而进一步筛选 Live Tail 会话。为此，请在选择日志流中，从下拉列表选择日志流的名称。或者，您可以使用选择日志流下的第二个框输入日志流名称前缀，然后将选择名称与前缀匹配的所有日志流。
5. （可选）要仅显示包含某些单词或其他字符串的日志事件，请在 Add filter patterns 中输入词或字符串。

例如，要仅显示包含词 Warning 的日志事件，请输入 **Warning**。筛选字段不区分大小写。您可以在该字段中包含多个术语和模式运算符：

- **error 404** 仅显示同时包含 error 和 404 的日志事件
- **?Error ?error** 显示包含 Error 或 error 的日志事件
- **-INFO** 显示不包括 INFO 的所有日志事件
- **{ \$.eventType = "UpdateTrail" }** 显示事件类型字段值为 UpdateTrail 的所有 JSON 日志事件

您也可以使用正则表达式 (regex) 进行筛选：

- **%ERROR%** 使用正则表达式显示由 ERROR 关键字组成的所有日志事件
- **{ \$.names = %Steve% }** 使用正则表达式显示属性 "name" 中包含 Steve 的 JSON 日志事件
- **[w1 = %abc%, w2]** 使用正则表达式显示第一个单词是 abc 的空格分隔日志事件

有关模式语法的更多信息，请参阅[筛选条件和模式语法](#)。

6. (可选) 要突出显示的某些日志事件，请在 Live Tail 下输入要搜索并突出显示的字词。逐一输入突出显示的字词。如果添加多个要突出显示的字词，则会为每个字词分配不同的颜色。突出显示指示器显示在包含指定字词的任意日志事件的左侧；当您在主窗口中展开日志事件以查看完整的日志事件时，突出显示指示器也会出现在字词本身的下方。

您可以搭配使用筛选和突出显示来快速排查问题。例如，您可以筛选事件以仅显示包含 `Error` 的事件，然后突出显示包含 `404` 的事件。

7. 要启动会话，请选择应用筛选条件

匹配的日志事件开始出现在窗口中。并将显示以下信息：

- 计时器显示 Live Tail 会话处于活动状态的时间。
 - `events/sec` 显示每秒有多少提取的日志事件与您设置的筛选条件相匹配。
 - 为了防止会话滚动得太快（因为许多事件都与筛选条件相匹配），CloudWatch Logs 可能只显示部分匹配事件。如果发生这种情况，屏幕上显示的匹配事件的百分比将以 % 已显示呈现。
8. 要暂停事件流以查看当前显示的内容，请单击事件窗口中的任意位置。
 9. 在会话期间，您可以使用以下内容来查看有关每个日志事件的更多详细信息。
 - 要在主窗口中显示日志事件的完整文本，请选择该日志事件旁边的箭头。
 - 要在侧窗口中显示日志事件的完整文本，请选择该日志事件旁边的 + 放大镜。事件流暂停，侧窗口出现。

在侧窗口中显示日志事件文本对于将其文本与主窗口中的其他事件进行比较很有用。

10. 要停止 Live Tail 会话，请选择停止。
11. 要重新启动会话，可以选择使用筛选条件面板来修改筛选条件，然后选择应用筛选条件。然后选择 Start (开始)。

跨账户跨区域日志集中化

Amazon CloudWatch Logs 数据集中化可结合 Amazon Organizations 使用，通过使用跨账户和跨区域集中化规则将多个成员账户中的日志数据收集到一个数据存储库中。您可以定义自动将日志数据从多个账户和 Amazon Web Services 区域复制到组织内集中账户的规则。此功能简化了日志整合流程，可在整个 Amazon 基础设施范围内实现更高效的集中监控、分析与合规性管理。

CloudWatch Logs 数据集中化提供了配置灵活性，可满足运营和安全要求，例如能够在目标账户中设置规则期间配置备份区域，从而确保增强恢复能力。此外，您可以完全控制从源账户复制的日志组的加密行为，以处理最初使用客户管理型 KMS 密钥加密的数据。

Note

CloudWatch Logs 集中化仅处理在您创建集中化规则后到达源账户的新日志数据。历史日志数据（创建规则之前存在的日志）不会进行集中。

数据集中化概念

在开始使用 CloudWatch Logs 数据集中化之前，请先熟悉以下概念：

集中化规则

一种用于定义如何将源账户和区域中的日志数据复制到目标账户和区域的配置。规则指定源条件和目标设置。

源账户

日志数据来源的 Amazon 账户。根据您定义的集中化规则，源账户中的日志事件将复制到目标账户。

目标账户

存储复制的日志数据的目标 Amazon 账户。该账户用作日志分析和监控的集中位置。

备份区域

目标账户中的可选辅助区域，可以在其中复制日志数据，以提高恢复能力并实现灾难恢复。

CloudWatch Logs 中的加密

日志组数据始终在 CloudWatch Logs 中加密。默认情况下，CloudWatch Logs 使用具有 256 位高级加密标准 Galois/Counter 模式（AES-GCM）的服务器端加密来对静态日志数据进行加密。

此外，您也可以使用 Amazon Key Management Service 进行此加密。如果这样做，则将使用 Amazon 拥有的 KMS 密钥或客户管理型 KMS 密钥进行加密。在创建日志组时或者在日志组已存在的情况下，通过将 KMS 密钥与日志组相关联，在日志组级别启用使用 Amazon KMS 进行 KMS 密钥加密。将 KMS 密钥与日志组关联后，该日志组所有新摄取的数据都将使用该密钥加密。此数据在其保留期内始终以加密格式存储。CloudWatch Logs 可随时根据请求解密该数据。当请求加密的数据时，例如对源账户运行日志集中化规则时，CloudWatch Logs 必须具有 KMS 密钥的权限。如果您使用的是客户管理型 KMS 密钥，请使用标签 `LogsManaged = true` 更新与源和目标日志组关联的 KMS 密钥。有关更多信息，请参阅《Amazon Key Management Service 开发人员指南》中的 [Amazon KMS 密钥](#)。

设置日志集中化

要设置 CloudWatch Logs 集中化，您需要配置集中化规则来定义日志数据如何从源账户中的日志组流向目标账户中的日志组。

启用集中化规则并将日志事件复制到目标账户后，您可以针对集中式日志组创建指标、订阅和账户筛选条件，并增强筛选功能。这些筛选条件可以针对来自特定源账户和区域的日志事件，并且可以将源账户和区域信息作为指标维度发出。有关更多信息，请参阅 [使用筛选条件从日志事件创建指标](#)。

先决条件

- 必须设置 Amazon Organizations，并且源账户和目标账户都必须属于该组织。
- 必须为 CloudWatch、管理账户和目标账户启用可信访问，以便提供对日志数据的访问权限。

创建集中化规则

请按照以下过程创建一个将日志数据从源账户复制到目标账户的集中化规则。

创建集中化规则

1. 以组织的管理账户或委派管理员账户身份导航至 CloudWatch 控制台。
2. 选择设置。
3. 导航到组织选项卡。
4. 选择配置规则。
5. 通过设置下面的字段来指定源详细信息，然后选择下一步：
 - a. 集中化规则名称：为集中化规则输入一个唯一名称。

b. 源账户：定义源选择标准，以选取要从中集中遥测数据的账户。选择标准可包括：

- 组织内的成员账户列表
- 组织内的组织单元列表
- 整个组织

您可以通过两种模式提供选择标准：

- 生成器：基于点击的体验，可生成源选择标准
- 编辑器：提供源选择标准的自由格式的文本框

源选择标准支持的语法：

- 支持的键：OrganizationId | OrganizationUnitId | AccountId | *
- 支持的运算符：= | IN | OR

c. 源区域：选择一个区域列表以查找要集中的遥测数据。

6. 通过设置下面的字段来指定目标详细信息，然后选择下一步：

- a. 目标账户：在组织中选择一个充当遥测数据中心目标的账户。
- b. 目标区域：选择一个存储集中式遥测数据副本的主要区域。
- c. 备份区域：根据需要选择存储集中式遥测数据的第二个副本的区域。

7. 通过设置下面的字段来指定遥测数据，然后选择下一步：

a. 日志组：选择以下选项之一：

- 所有日志组：集中源账户中所有日志组的日志。
- 筛选日志组：根据日志组选择标准集中源账户中一部分日志组的日志。您可以通过两种模式提供选择标准：
 - 生成器：基于点击的体验，可生成日志组选择标准
 - 编辑器：提供日志组选择标准的自由格式的文本框

日志组选择标准支持的语法：

- 支持的键：LogGroupName | *
- 支持的运算符：= | != | IN | NOT IN | AND | OR | LIKE | NOT LIKE

b. KMS 加密的日志组

 Important

如果集中化规则中提供的 KMS 密钥不允许 CloudWatch Logs 使用它，则 CloudWatch 集中化规则无法将日志从源账户传输到目标日志组。有关更多信息，请参阅 [步骤 2：设置 KMS 密钥的权限](#)。

请选择以下选项之一：

- 不要集中使用客户管理型 KMS 密钥加密的日志组：跳过集中使用客户管理型 KMS 密钥加密的源日志组中的日志事件。
- 使用 Amazon 托管式 KMS 密钥在目标账户中集中通过客户管理型 KMS 密钥加密的日志组：将通过客户管理型 KMS 密钥加密的源日志组中的日志事件集中到未与客户管理型 KMS 密钥关联而是使用 Amazon 托管式 KMS 密钥的目标日志组中。

选择此设置后，您还必须设置以下内容：

- 目标加解密密钥 ARN：属于目标账户和主目标区域的 KMS 密钥的 ARN，与新创建的目标日志组关联。
- 备份目标加解密密钥 ARN（可选）：属于目标账户和备份目标区域的 KMS 密钥的 ARN，与新创建的目标日志组关联。

 Note

请注意，此设置仅适用于使用客户管理型 KMS 密钥加密源日志组的情况，并且仅适用于目标账户中新创建的日志组。

8. 查看集中化规则，根据需要进行任何最后的编辑，然后选择创建集中化策略。

修改集中化规则

请按照以下过程修改现有的集中化规则。

修改集中化规则

1. 以组织的管理账户或委派管理员账户身份导航至 CloudWatch 控制台。
2. 选择设置。

3. 导航到组织选项卡。
4. 选择管理规则。
5. 选择要更新规则，然后选择编辑。
6. 根据需要更新规则配置，然后选择下一步继续执行每个步骤。
7. 在步骤 4“查看和配置”中，选择更新集中化策略。

查看集中化规则

请按照以下过程查看现有集中化规则的详细信息。

查看集中化规则

1. 以组织的管理账户或委派管理员账户身份导航至 CloudWatch 控制台。
2. 选择设置。
3. 导航到组织选项卡。
4. 选择管理规则。
5. 查看所有现有集中化规则的列表，然后选择一个特定规则名称以查看其详细信息。

删除集中化规则

请按照以下过程删除现有的集中化规则。

删除集中化规则

1. 以组织的管理账户或委派管理员账户身份导航至 CloudWatch 控制台。
2. 选择设置。
3. 导航到组织选项卡。
4. 选择管理规则。
5. 选择要删除的规则，然后选择删除。
6. 确认删除并选择删除。

监控集中化

您可以使用 CloudWatch 指标、CloudWatch Logs 控制台和 Amazon CloudTrail 日志来监控集中化规则的状态和性能。这有助于确保日志数据成功复制，并识别集中化配置中的任何问题。

在控制台中监控集中化

使用 CloudWatch Logs 控制台查看集中化规则的状态和活动。

在控制台中监控集中化规则

1. 以组织的管理账户或委派管理员账户身份导航至 CloudWatch 控制台。
2. 选择设置。
3. 导航到组织选项卡。
4. 选择管理规则。
5. 查看集中化规则列表，其中显示：
 - 规则名称：每个集中化规则的名称
 - 规则状态：当前操作状态（活动、非活动、错误）
 - 创建日期：创建规则的时间
 - 目标账户 ID：目标账户的账户 ID
 - 目标区域：目标账户的区域
6. 选择一个特定规则名称以查看规则配置详细信息

集中化监控

您可以使用控制台界面和 API 操作来监控集中化规则。

当前监控功能包括：

- 规则运行状况：通过控制台或 GetCentralizationRuleForOrganization API 监控集中化规则的整体运行状况
- 规则配置：查看规则设置和上次更新时间戳
- 失败原因：当规则被标记为 UNHEALTHY 时，查看详细的失败信息
- API 活动：通过 CloudTrail 日志跟踪集中化 API 调用

监控规则运行状况

每个集中化规则都有运行状况指示其是否正常运行。您可以通过控制台或使用 API 以编程方式检查规则运行状况。

规则运行状况包括：

- **HEALTHY**：规则运行正常，并按配置复制日志数据
- **UNHEALTHY**：规则遇到问题，可能无法正确复制数据
- **PROVISIONING**：组织的集中化正在设置过程中。

当规则被标记为 **UNHEALTHY** 时，**FailureReason** 字段会提供有关需要解决的具体问题的详细信息。

使用 Amazon CloudTrail 监控集中化 API 调用

Amazon CloudTrail 会记录对集中化服务的 API 调用，使您可以跟踪配置更改并排查 Amazon Organizations 成员账户的问题。

关键 CloudTrail 集中化事件包括：

- **CreateCentralizationRuleForOrganization**：创建新的集中化规则时
- **UpdateCentralizationRuleForOrganization**：修改现有规则时
- **DeleteCentralizationRuleForOrganization**：删除规则时
- **GetCentralizationRuleForOrganization**：检索规则详细信息时
- **ListCentralizationRulesForOrganization**：列出规则时

您可以使用 CloudTrail 日志来审核集中化配置更改，并将其与性能问题或复制失败关联起来。

使用日志组和日志流

日志流是共享同一个源的一系列日志事件。CloudWatch Logs 中每个独立的日志源构成一个独立的日志流。

日志组是一组具有相同保留期、监控和访问控制设置的日志流。您可以定义日志组并指定向各组中放入哪些流。对可属于一个日志组的日志流数没有限制。

对于需要整合来自多个账户和区域的日志数据的组织，您可以使用 CloudWatch Logs 集中化自动将日志组复制到中央账户。有关更多信息，请参阅 [跨账户跨区域日志集中化](#)。

您可以使用本部分中的过程处理日志组和日志流。

在 CloudWatch Logs 中创建日志组

使用《Amazon CloudWatch Logs 用户指南》前面章节中的步骤在 Amazon EC2 实例上安装 CloudWatch Logs 代理时，该过程中也将创建日志组。您还可以直接在 CloudWatch 控制台中创建日志组。

创建日志组

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择日志组。
3. 选择操作，然后选择创建日志组。
4. 输入日志组的名称，然后选择创建日志组。

Tip

您可以通过导航窗格中的 Favorites and recents (收藏夹和最近记录) 菜单收藏日志组以及控制面板和警报。在 Recently visited (最近访问) 列中，将鼠标悬停在您想要收藏的日志组上，然后选择旁边的星形符号。

将日志发送到日志组

CloudWatch Logs 会自动接收来自多个 Amazon 服务的日志事件。您也可以使用以下方法之一将其他日志事件发送到 CloudWatch Logs：

- CloudWatch 代理 – 统一的 CloudWatch 代理可以同时将指标和日志发送到 CloudWatch Logs。有关安装和使用 CloudWatch 代理的信息，请参阅《Amazon CloudWatch 用户指南》中的[使用 CloudWatch 代理从 Amazon EC2 实例和本地服务器中收集指标和日志](#)。
- Amazon CLI – [put-log-events](#) 会将批量日志事件上载到 CloudWatch Logs。
- 以编程方式 – [PutLogEvents](#) API 使您能够以编程方式将批量日志事件上载到 CloudWatch Logs。

查看发送到 CloudWatch Logs 的日志数据

您可以查看和滚动浏览 CloudWatch Logs 代理发送到 CloudWatch Logs 的每一个日志流的日志数据。您可以指定要查看的日志数据的时间范围。

查看日志数据

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Log groups (日志组) 。
3. 对于 Log Groups (日志组) ，选择日志组以查看日志流。
4. 在日志组列表中，选择要查看的日志组的名称。
5. 从日志流列表中，选择要查看的日志流的名称。
6. 要更改日志数据的显示方式，请执行下列操作之一：
 - 要展开单个日志事件，请选择该日志事件旁边的箭头。
 - 要展开所有日志事件并以纯文本形式查看它们，请在日志事件列表上方选择 Text。
 - 要筛选日志事件，请在搜索字段中键入所需的搜索筛选条件。有关更多信息，请参阅[使用筛选条件从日志事件创建指标](#)。
 - 要查看指定日期和时间范围的日志数据，请在搜索筛选条件旁边，选择日期和时间旁的箭头。要指定日期和时间范围，请选择 Absolute (绝对) 。要选择预定义的分钟数、小时数、天数或周数，请选择 Relative (相对) 。还可以在 UTC 和本地时区之间切换。

使用筛选条件模式搜索日志数据

可以使用 [指标筛选条件](#)、[订阅筛选条件](#)、[筛选日志事件](#)和 [Live Tail 的筛选条件模式语法](#) 搜索日志数据。您可以搜索日志组中的所有日志流，也可以使用 Amazon CLI 搜索特定日志流。每次搜索运行时，都会返回到所找到数据的第一页，并且使用令牌来检索下一页数据或继续搜索。如果没有返回任何结果，可以继续搜索。

您可以设置要查询的时间范围来限制搜索范围。您可以从较大范围开始，看看感兴趣的日志行在何处，然后缩短时间范围在相应时间范围内查看您感兴趣的日志。

您还可以通过从日志提取的指标直接定向至相应日志。

如果您登录了在 CloudWatch 跨账户可观察性中设置为监控账户的账户，则可以从与该监控账户关联的源账户搜索和筛选日志事件。有关更多信息，请参阅 [CloudWatch 跨账户可观察性](#)。

使用控制台搜索日志条目

您可以使用控制台搜索满足指定条件的日志条目。

使用控制台搜索日志

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Log groups (日志组)。
3. 对于 Log Groups，选择包含要搜索的日志流的日志组的名称。
4. 对于 Log Streams，选择要搜索的日志流的名称。
5. 在 Log events (日志事件) 下，输入要使用的筛选条件语法。

使用控制台搜索某个时间范围的所有日志条目

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Log groups (日志组)。
3. 对于 Log Groups，选择包含要搜索的日志流的日志组的名称。
4. 选择 Search log group (搜索日志组)。
5. 对于 Log events (日志事件)，请选择日期和时间范围，然后输入筛选条件语法。

使用 Amazon CLI 搜索日志条目

您可以使用 Amazon CLI 搜索满足指定条件的日志条目。

使用 Amazon CLI 搜索日志条目

在命令提示符处，运行以下 [filter-log-events](#) 命令。使用 `--filter-pattern` 将结果限制为指定的筛选条件模式，并使用 `--log-stream-names` 将结果限制为指定的日志流。

```
aws logs filter-log-events --log-group-name my-group [--log-stream-  
names LIST_OF_STREAMS_TO_SEARCH] [--filter-pattern VALID_METRIC_FILTER_PATTERN]
```

使用 Amazon CLI 搜索给定时间范围内的日志条目

在命令提示符处，运行以下 [filter-log-events](#) 命令：

```
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]
```

从指标定向至日志

您可以从控制台的其他部分转到特定的日志条目。

从控制面板小部件转到日志

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Dashboards (控制面板)。
3. 选择控制面板。
4. 在小部件上，选择 View logs 图标，然后选择 View logs in this time range。如果存在多个指标筛选条件，请从列表选择一个。如果有更多指标筛选条件，超出列表中可以显示的数量，请选择 More metric filters，然后选择或搜索指标筛选条件。

从指标转到日志

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Metrics (指标)。
3. 在 All metrics (所有指标) 选项卡上的搜索字段中，键入指标的名称，然后按 Enter。
4. 从搜索结果中选择一个或多个指标。
5. 选择 Actions (操作)、View logs (查看日志)。如果存在多个指标筛选条件，请从列表选择一个。如果有更多指标筛选条件，超出列表中可以显示的数量，请选择 More metric filters，然后选择或搜索指标筛选条件。

故障排除

搜索耗时太长

如果有大量日志数据，搜索可能需要很长时间才能完成。要提高搜索速度，可以执行以下操作：

- 如果使用 Amazon CLI，则可以将搜索限制为仅搜索您感兴趣的日志流。例如，如果日志组有 1000 个日志流，但您只需要查看三个已知相关的日志流，则可以使用 Amazon CLI 将搜索限制为仅搜索相应日志组中的那三个日志流。
- 使用更短、更细粒度的时间范围，从而减少搜索的数据量和加快查询速度。

更改 CloudWatch Logs 中的日志数据留存

默认情况下，日志数据将无限期存储在 CloudWatch Logs 中。但是，您可以配置要在日志组中存储日志数据多长时间。存储时间超过当前保留期设置的所有数据将被删除。您可以随时更改每个日志组的日志保留期。

Note

当日志事件达到保留设置时，CloudWatch Logs 不会立即删除它们。此后通常需要 72 小时才能删除日志事件，但在极少数情况下可能需要更长时间。

这意味着，如果将日志组更改为具有更长的保留期设置，而该日志组包含已过期但实际上尚未删除的日志事件，则在达到新保留日期后，这些日志事件最多需要 72 小时才能被删除。要确保永久删除日志数据，请将日志组保留在较低的保留期设置，直到上一个保留期结束后 72 小时过去，或者您确认已删除较早的日志事件。

当日志事件达到其保留设置时，这些日志事件将被标记为待删除。在被标记为待删除后，即使它们要等到以后才真正被删除，也不会再增加您的档案存储成本。当您使用 API 检索 `storedBytes` 值以查看日志组存储了多少字节时，这些标记为待删除的日志事件也不包括在内。

更改日志保留期设置

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Logs (日志) 和 Log groups (日志组)。
3. 找到要更新的日志组。
4. 在该日志组的保留列中，选择当前保留期设置，例如 Never Expire。
5. 在保留设置中，对于事件过期时间，选择日志保留值，然后选择保存。

在 Amazon CloudWatch Logs 中标记日志组

您可以将自己的元数据分配给您在 Amazon CloudWatch Logs 中以标签形式创建的日志组。标签是您为日志组定义的键值对。使用标签是管理 Amazon 资源和组织数据（包括账单数据）的一种简单却强有力的方式。

Note

您可以使用标签控制对 CloudWatch Logs 资源（包括日志组和目标）的访问。由于日志组和日志流之间存在分层关系，因此在日志组级别控制对日志流的访问。有关使用标签控制访问的更多信息，请参阅[使用标签控制对 Amazon Web Services 资源的访问](#)。

目录

- [有关标签的基本知识](#)
- [使用标签跟踪成本](#)
- [标签限制](#)
- [使用 Amazon CLI 标记日志组](#)
- [使用 CloudWatch Logs API 标记日志组](#)

有关标签的基本知识

可以使用 Amazon CloudFormation 或 Amazon CLI 或 CloudWatch Logs API 完成以下任务：

- 在创建日志组时向其添加标签。
- 向现有日志组添加标签。
- 列出日志组的标签。
- 从日志组删除标签。

您可以使用标签对日志组进行分类。例如，您可以按用途、所有者或环境对它们进行分类。由于您定义每个标签的键和值，因此您可以创建一组自定义类别来满足您的特定需求。例如，您可以定义一组标签来帮助您按所有者和关联应用程序跟踪日志组。以下几个标签示例：

- 项目：项目名称

- 所有者：名称
- 用途：负载测试
- 应用程序：应用程序名称
- 环境：生产

使用标签跟踪成本

您可以使用标签对 Amazon 成本进行分类和跟踪。当您将标签应用于 Amazon 资源（包括日志组）时，您的 Amazon 成本分配报告将包括按标签汇总的使用率和成本。您可以设置代表业务类别（例如成本中心、应用程序名称或所有者）的标签，以便整理多种服务的成本。有关更多信息，请参阅 Amazon Billing 用户指南中的[对自定义账单报告使用成本分配标签](#)。

标签限制

以下限制适用于标签。

基本限制

- 每个日志组的最大标签数为 50。
- 标签键和值区分大小写。
- 无法更改或编辑已删除日志组的标签。

标签键限制

- 每个标签键必须是唯一的。如果您添加的标签具有已使用的键，则您的新标签将覆盖现有键值对。
- 标签键不能以 aws: 开头，因为此前缀将预留以供 Amazon 使用。Amazon 将代表您创建以此前缀开头的标签，但您不能编辑或删除这些标签。
- 标签键的长度必须介于 1 和 128 个 Unicode 字符之间。
- 标签键必须包含以下字符：Unicode 字母、数字、空格和以下特殊字符：_ . / = + - @。

标签值限制

- 标签值的长度必须介于 0 和 255 个 Unicode 字符之间。
- 标签值可以为空。另外，它们必须包含以下字符：Unicode 字母、数字、空格和以下任意特殊字符：_ . / = + - @。

使用 Amazon CLI 标记日志组

您可以使用 Amazon CLI 添加、列出和删除标签。有关示例，请参阅以下文档：

[create-log-group](#)

创建日志组。您可以选择在创建日志组时添加标签。

[- tag-resource](#)

将一个或多个标签（键值对）分配给指定的 CloudWatch Logs 资源。

[list-tags-for-resource](#)

显示与 CloudWatch Logs 资源关联的标签。

[- untag-resource](#)

从指定的 CloudWatch Logs 资源中删除一个或多个标签。

使用 CloudWatch Logs API 标记日志组

您可以使用 CloudWatch Logs API 添加、列出和删除标签。有关示例，请参阅以下文档：

[CreateLogGroup](#)

创建日志组。您可以选择在创建日志组时添加标签。

[TagResource](#)

将一个或多个标签（键值对）分配给指定的 CloudWatch Logs 资源。

[ListTagsForResource](#)

显示与 CloudWatch Logs 资源关联的标签。

[UntagResource](#)

从指定的 CloudWatch Logs 资源中删除一个或多个标签。

使用 Amazon Key Management Service 对 CloudWatch Logs 中的日志数据进行加密

日志组数据始终在 CloudWatch Logs 中加密。默认情况下，CloudWatch Logs 使用具有 256 位高级加密标准 Galois/Counter 模式（AES-GCM）的服务器端加密来对静态日志数据进行加密。此外，您也可

以使用 Amazon Key Management Service 进行这一加密。如果这样做，则使用 Amazon KMS 密钥进行加密。在创建日志组时或者在日志组已存在的情况下，通过将 KMS 密钥与日志组相关联，在日志组级别启用使用 Amazon KMS 加密。

 Important

CloudWatch Logs 现在支持使用 `kms:EncryptionContext:aws:logs:arn` 作为密钥和使用日志组的 ARN 作为该密钥的值来加密上下文。如果您的日志组已使用 KMS 密钥加密，并且您希望将密钥限制在单个账户和日志组中使用，则应分配一个新的 KMS 密钥，该 KMS 密钥在 IAM policy 中包含一个条件。有关更多信息，请参阅 [Amazon KMS 密钥和加密上下文](#)。

 Important

CloudWatch Logs 现在支持 `kms:ViaService`，其允许日志代表您进行 Amazon KMS 调用。您应该将其添加到密钥策略或 IAM 中调用 CloudWatch Logs 的角色中。有关更多信息，请参阅 [kms:ViaService](#)。

将 KMS 密钥与日志组关联后，该日志组所有新摄取的数据都将使用该密钥加密。此数据在其保留期内始终以加密格式存储。CloudWatch Logs 可随时根据请求解密该数据。无论何时请求加密数据，CloudWatch Logs 都必须拥有对 KMS 的权限。

如果您之后取消了 KMS 密钥与日志组的关联，CloudWatch Logs 将使用 CloudWatch Logs 默认加密方法来加密新摄入的数据。所有使用 KMS 加密的先前摄取的数据仍使用 KMS 加密。取消关联 KMS 密钥后，CloudWatch Logs 仍然可以返回该数据，这是因为 CloudWatch Logs 仍然可以继续引用该密钥。但如果该密钥之后被禁用，则 CloudWatch Logs 无法读取使用该密钥加密的日志。

 Important

CloudWatch Logs 仅支持对称 KMS 密钥。请勿使用非对称密钥对日志组中的数据进行加密。有关更多信息，请参阅 [使用对称和非对称密钥](#)。

限制

- 要执行下列步骤，您必须具有以下权限：`kms:CreateKey`、`kms:GetKeyPolicy` 和 `kms:PutKeyPolicy`。

- 将密钥与日志组关联或解除关联后，最多可能需要五分钟时间，此操作才能生效。
- 如果撤消 CloudWatch Logs 对关联密钥的访问权限或删除关联的 KMS 密钥，您将无法再检索 CloudWatch Logs 中的加密数据。
- 您无法使用 CloudWatch 控制台将 KMS 密钥与现有日志组关联。

步骤 1：创建 Amazon KMS 密钥

要创建 KMS 密钥，请使用以下 [create-key](#) 命令：

```
aws kms create-key
```

输出包含密钥的密钥 ID 和 Amazon 资源名称 (ARN)。下面是示例输出：

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

步骤 2：设置 KMS 密钥的权限

默认情况下，所有 Amazon KMS 密钥都是私有的。只有资源所有者可以使用它来加密和解密数据。但是，资源拥有者可以将 KMS 密钥的访问权限授予其他用户和资源。通过此步骤，您将授予 CloudWatch Logs 服务主体和调用方角色使用该密钥的权限。此服务主体必须位于存储 KMS 密钥的相同 Amazon 区域内。

作为最佳做法，我们建议您将 KMS 密钥的使用限制为仅限于您指定的 Amazon 账户或日志组。

首先，使用以下 `policy.json` 命令将您的 KMS 密钥默认策略保存为：

```
aws kms get-key-policy --key-id key-id --policy-name default --output text > ./policy.json
```

在文本编辑器中打开 `policy.json` 文件，然后从以下语句之一中添加粗体的部分。使用逗号将现有语句与新语句分隔。这些语句使用 Condition 部分来增强 Amazon KMS 密钥的安全性。有关更多信息，请参阅 [Amazon KMS 密钥和加密上下文](#)。

本示例中的 Condition 部分将密钥限制为单个日志组 ARN。

JSON

```
{
  "Version": "2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:Describe*"
      ],
      "Resource": "*",
      "Condition": {
```

```

        "ArnEquals": {
            "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:111122223333:log-group:log-group-name"
        }
    }
}

```

本示例中的 Condition 部分对 Amazon KMS 密钥的使用进行限制，只有指定账户才能使用该密钥，但该密钥可用于任何日志组。

JSON

```

{
  "Version": "2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:Describe*"
      ],
      "Resource": "*",
      "Condition": {

```

```

        "ArnLike": {
            "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:123456789012:*"
        }
    }
}
]
}

```

接下来，为将要调用 CloudWatch Logs 的角色添加权限。您可以通过在 Amazon KMS 密钥策略中添加额外语句或通过 IAM 对角色本身执行此操作。CloudWatch Logs 使用 `kms:ViaService` 代表客户调用 Amazon KMS。有关更多信息，请参阅 [kms:ViaService](#)。

要在 Amazon KMS 密钥策略中添加权限，请将以下语句添加到您的密钥策略中。如果使用此方法，最佳实践是将策略范围限定于将与 Amazon KMS 加密日志组交互的角色。

```

{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::account_id:role/role_name"
  },
  "Action": [
    "kms:Encrypt",
    "kms:ReEncrypt*",
    "kms:Decrypt",
    "kms:GenerateDataKey*",
    "kms:Describe*"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": [
        "logs.region.amazonaws.com"
      ]
    }
  }
}

```

或者，如果您想在 IAM 中管理角色权限，则可以通过以下策略添加等效权限。您可以将此策略添加到现有角色策略中，也可以将其作为单独的额外策略附加到角色。如果您使用此方法，最佳实践是将策略范围限定于仅用于日志加密的 Amazon KMS 密钥。有关更多信息，请参阅[编辑 IAM 策略](#)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:ReEncrypt*",
        "kms:Decrypt",
        "kms:GenerateDataKey*",
        "kms:Describe*"
      ],
      "Condition": {
        "StringEquals": {
          "kms:ViaService": [
            "logs.us-east-1.amazonaws.com"
          ]
        }
      },
      "Resource": "arn:aws:kms:us-east-1:444455556666:key/key_id"
    }
  ]
}
```

最后，使用以下 [put-key-policy](#) 命令添加更新后的策略：

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

步骤 3：将日志组与 KMS 密钥关联

可以在创建日志组时或创建完成后将 KMS 密钥与它关联。

要确定日志组是否已经关联了 KMS 密钥，请使用以下 [describe-log-groups](#) 命令：

```
aws logs describe-log-groups --log-group-name-prefix "log-group-name-prefix"
```

如果输出包含 kmsKeyId 字段，则日志组将与该字段值所对应的键相关联。

在创建日志组时将 KMS 密钥与它关联

按如下方式使用 [create-log-group](#) 命令：

```
aws logs create-log-group --log-group-name my-log-group --kms-key-id "key-arn"
```

将 KMS 密钥与现有日志组关联

按如下方式使用 [associate-kms-key](#) 命令：

```
aws logs associate-kms-key --log-group-name my-log-group --kms-key-id "key-arn"
```

步骤 4：取消日志组与密钥的关联

要取消 KMS 密钥与日志组的关联，请使用以下 [disassociate-kms-key](#) 命令：

```
aws logs disassociate-kms-key --log-group-name my-log-group
```

Amazon KMS 密钥和加密上下文

为了增强 Amazon Key Management Service 密钥和加密日志组的安全性，CloudWatch Logs 现在将日志组 ARN 作为用于加密日志数据的加密上下文的一部分。加密上下文是一组用作附加的经身份验证的数据的键值对。通过加密上下文，您可以使用 IAM policy 条件按 Amazon 账户和日志组来限制对 Amazon KMS 密钥的访问权限。有关更多信息，请参阅[加密上下文](#)和 [IAM JSON 策略元素：条件](#)。

我们建议您为每个加密的日志组使用不同的 KMS 密钥。

如果您有以前加密的日志组，但现在希望将日志组更改为使用仅适用于该日志组的新 KMS 密钥，请按照下列步骤操作。

将加密的日志组转换为使用某个 KMS 密钥，并且有策略将此 KMS 密钥设置为仅限于该日志组使用

1. 输入以下命令以查找该日志组的当前密钥的 ARN：

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

输出包括以下行。记录 ARN。您需要在步骤 7 中使用它。

```
...
"kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/01234567-89ab-
cdef-0123-456789abcdef"
...
```

2. 输入以下命令以创建新的 KMS 密钥：

```
aws kms create-key
```

3. 输入以下命令以将新密钥的策略保存到 policy.json 文件中：

```
aws kms get-key-policy --key-id new-key-id --policy-name default --output text > ./
policy.json
```

4. 使用文本编辑器打开 policy.json 并向策略添加 Condition 表达式：

JSON

```
{
  "Version": "2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
```



```

        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:Describe*"
    ],
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:111122223333:log-group:LOG-GROUP-NAME"
        }
    }
}
]
}

```

5. 输入以下命令以将更新的策略添加到新的 KMS 密钥：

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

6. 输入以下命令以将策略与日志组关联：

```
aws logs associate-kms-key --log-group-name my-log-group --kms-key-id new-key-ARN
```

CloudWatch Logs 现在使用新的密钥加密所有新数据。

7. 接下来，从旧的密钥撤除 Decrypt 之外的所有其他权限。首先，输入以下命令以检索旧策略：

```
aws kms get-key-policy --key-id old-key-ARN --policy-name default --output text
> ./policy.json
```

8. 使用文本编辑器打开 policy.json 并从 Action 列表中删除所有值，但 kms:Decrypt 除外
- JSON

```

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

```

```
    "Principal": {
      "AWS": "arn:aws:iam::111122223333:root"
    },
    "Action": "kms:*",
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Principal": {
      "Service": "logs.region.amazonaws.com"
    },
    "Action": [
      "kms:Decrypt"
    ],
    "Resource": "*"
  }
]
```

9. 输入以下命令以将更新的策略添加到旧的密钥：

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

通过屏蔽帮助保护敏感的日志数据

可以使用日志组数据保护策略帮助保护 CloudWatch Logs 提取的敏感数据。使用这些策略可以审核和屏蔽账户中日志组提取的日志事件中出现的敏感数据。

创建数据保护策略时，默认情况下，将在所有出口点屏蔽与您所选数据标识符相匹配的敏感数据，包括 CloudWatch Logs Insights、指标筛选条件和订阅筛选条件。只有拥有 logs:Unmask IAM 权限的用户才能查看未屏蔽的数据。

您可以为账户中所有日志组创建数据保护策略，也可以为各个日志组创建数据保护策略。当为整个账户创建策略时，它既适用于现有日志组，也适用于将来创建的日志组。

如果为整个账户创建了数据保护策略，并且还单个日志组创建了策略，则这两个策略都适用于该日志组。任一策略中指定的所有托管数据标识符都会在该日志组中进行审核和屏蔽。

 Note

仅标准日志类中的日志组支持屏蔽敏感数据。如果您为账户中的所有日志组创建数据保护策略，则该策略仅适用于标准日志类中的日志组。有关日志类的更多信息，请参阅 [日志类](#)。

每个日志组只能有一个日志组级别的数据保护策略，但是该策略可以指定许多要审核和屏蔽的托管数据标识符。数据保护策略的限制为 30,720 个字符。

 Important

将敏感数据摄入日志组时会进行检测并屏蔽。设置数据保护策略时，不会屏蔽在该时间之前摄入到日志组的日志事件。

CloudWatch Logs 支持许多托管数据标识符，它们提供您可以选择的预配置数据类型来保护财务数据、个人健康信息 (PHI) 和个人身份信息 (PII)。CloudWatch Logs 数据保护允许您利用模式匹配和机器学习模型来检测敏感数据。对于某些类型的托管数据标识符，检测还取决于是否找到与敏感数据接近的某些关键字。您还可以使用自定义数据标识符来创建针对特定使用场景量身定制的数据标识符。

当检测到与您选择的数据标识符匹配的敏感数据时，系统会向 CloudWatch 发出指标。这是 LogEventsWithFindings 指标，它是在 AWS/Logs 命名空间中发出的。您可以使用此指标创建 CloudWatch 警报，并且可以在图表和仪表板中将其可视化。数据保护发出的指标是出售的指标，是免费的。有关 CloudWatch Logs 发送给 CloudWatch 的指标的更多信息，请参阅 [使用 CloudWatch 指标进行监控](#)。

每个托管数据标识符都设计用于检测特定类型的敏感数据，例如信用卡号、Amazon 秘密访问密钥或者特定国家或地区的护照号码。创建数据保护策略时，您可以将其配置为使用这些标识符来分析通过日志组摄取的日志，并在检测到敏感数据时采取措施。

CloudWatch Logs 数据保护可以使用托管数据标识符检测以下类别的敏感数据：

- 凭证，例如私有密钥或 Amazon 秘密访问密钥
- 财务信息，例如信用卡号
- 个人身份信息 (PII)，例如驾驶执照或社会安全号码
- 受保护健康信息 (PHI)，例如健康保险或医疗识别号码
- 设备标识符，例如 IP 地址或 MAC 地址

有关您可以保护的数据类型的详细信息，请参阅 [您可以保护的数据类型](#)。

目录

- [了解数据保护策略](#)
 - [什么是数据保护策略？](#)
 - [数据保护策略采用什么结构？](#)
 - [数据保护策略的 JSON 属性](#)
 - [策略语句的 JSON 属性](#)
 - [策略语句操作的 JSON 属性](#)
- [创建或使用数据保护策略所需的 IAM 权限](#)
 - [账户级数据保护策略所需的权限](#)
 - [单个日志组的数据保护策略所需的权限](#)
 - [数据保护策略示例](#)
- [创建账户范围的数据保护策略](#)
 - [控制台](#)
 - [Amazon CLI](#)
 - [Amazon CLI 或 API 操作的数据保护策略语法](#)
- [为单个日志组创建数据保护策略](#)
 - [控制台](#)
 - [Amazon CLI](#)
 - [Amazon CLI 或 API 操作的数据保护策略语法](#)
- [查看未屏蔽的数据](#)
- [审计结果报告](#)
 - [所需的密钥策略用于将审计调查结果发现发送给 Amazon KMS 保护的存储桶](#)
- [您可以保护的数据类型](#)
 - [敏感数据类型的 CloudWatch Logs 托管数据标识符](#)
 - [凭证](#)
 - [凭证数据类型的数据标识符 ARN](#)
 - [设备标识符](#)
 - [设备数据类型的数据标识符 ARN](#)

- [财务数据类型的数据标识符 ARN](#)
- [受保护健康信息 \(PHI \)](#)
 - [受保护健康信息 \(PHI, Protected Health Information\) 数据类型的数据标识符 ARN](#)
- [个人信息 \(PII \)](#)
 - [驾驶执照识别号的关键字](#)
 - [国民身份证号的关键字](#)
 - [护照号码的关键字](#)
 - [纳税人识别号和参考号的关键字](#)
 - [个人信息 \(PII, Personally Identifiable Information\) 的数据标识符 ARN](#)
- [自定义数据标识符](#)
 - [什么是自定义数据标识符？](#)
 - [自定义数据标识符限制](#)
 - [在控制台中使用自定义数据标识符](#)
 - [在数据保护策略中使用自定义数据标识符](#)

了解数据保护策略

主题

- [什么是数据保护策略？](#)
- [数据保护策略采用什么结构？](#)

什么是数据保护策略？

CloudWatch Logs 使用数据保护策略选择要扫描的敏感数据，以及为了保护这些数据而采取的操作。要选择感兴趣的敏感数据，您可以使用[数据标识符](#)。CloudWatch Logs 数据保护功能随后使用机器学习和模式匹配来检测敏感数据。要对找到的数据标识符采取操作，您可以定义审计和去身份识别操作。利用这些操作，您可以记录已找到（或未找到）的敏感数据，并在查看日志事件时屏蔽敏感数据。

数据保护策略采用什么结构？

如下图所示，数据保护策略文档包含以下元素：

- 文档顶部的可选策略范围信息
- 一个定义审计和去身份识别操作的语句

每个 CloudWatch Logs 日志组只能定义一个数据保护策略。数据保护策略可以有一个或多个拒绝或去身份识别语句，但只能有一个审计语句。

数据保护策略的 JSON 属性

数据保护策略需要以下基本策略信息用于识别：

- Name (名称) – 策略名称。
- Description (描述) (可选)– 策略描述。
- Version (版本) – 策略语言版本。当前版本为 2021-06-01。
- Statement (语句) – 指定数据保护策略操作的语句列表。

```
{
  "Name": "CloudWatchLogs-PersonalInformation-Protection",
  "Description": "Protect basic types of sensitive data",
  "Version": "2021-06-01",
  "Statement": [
    ...
  ]
}
```

策略语句的 JSON 属性

策略语句设置数据保护操作的检测上下文。

- Sid (可选) – 语句标识符。
- DataIdentifier – CloudWatch Logs 应扫描的敏感数据。例如，姓名、地址或电话号码。
- 操作 – 后续操作，即审计或去身份识别。CloudWatch Logs 在找到敏感数据时会执行这些操作。

```
{
  ...
  "Statement": [
    {
      "Sid": "audit-policy",
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/Address"
      ],
      "Operation": {
        "Audit": {
```

```
        "FindingsDestination": {}
    }
},
```

策略语句操作的 JSON 属性

策略语句设置以下数据保护操作之一。

- 审计 - 在不中断日志记录的情况下发出指标和结果报告。匹配的字符串会增加 CloudWatch Logs 发布到 CloudWatch 中 AWS/Logs 命名空间的 LogEventsWithFindings 指标。您可以使用这些指标创建警报。

有关结果报告的示例，请参阅 [审计结果报告](#)。

有关 CloudWatch Logs 发送给 CloudWatch 的指标的更多信息，请参阅 [使用 CloudWatch 指标进行监控](#)。

- 去身份识别 – 在不中断日志记录的情况下屏蔽敏感数据。

创建或使用数据保护策略所需的 IAM 权限

为了能够使用日志组的数据保护策略，您必须具有下表所示的特定权限。账户范围的数据保护策略和适用于单个日志组的数据保护策略的权限不同。

账户级数据保护策略所需的权限

 **Note**

如果您在 Lambda 函数内执行任何此类操作，则 Lambda 执行角色和权限边界还必须包含以下权限。

操作	所需的 IAM 权限	资源
创建没有审计目标的数据保护策略	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*

操作	所需的 IAM 权限	资源
将 CloudWatch Logs 作为审计目标来创建数据保护策略	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*
	logs:CreateLogDelivery	*
	logs:PutResourcePolicy	*
	logs:DescribeResourcePolicies	*
	logs:DescribeLogGroups	*
将 Firehose 作为审计目标来创建数据保护策略	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*
	logs:CreateLogDelivery	*
	firehose:TagDeliveryStream	arn:aws:logs:::deliverystream/ <i>YOUR_DELIVERY_STREAM</i>
将 Amazon S3 作为审计目标来创建数据保护策略	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*

操作	所需的 IAM 权限	资源
	logs:CreateLogDelivery	*
	s3:GetBucketPolicy	arn:aws:s3::: <i>YOUR_BUCKET</i>
	s3:PutBucketPolicy	arn:aws:s3::: <i>YOUR_BUCKET</i>
在指定日志组中取消屏蔽已屏蔽的日志事件	logs:Unmask	arn:aws:logs::log-group:*
查看现有数据保护策略	logs:GetDataProtectionPolicy	*
删除数据保护策略	logs>DeleteAccountPolicy	*
	logs>DeleteDataProtectionPolicy	*

如果已经将任何数据保护审计日志发送到目标，则将日志发送到同一目标的其他策略只需要 logs:PutDataProtectionPolicy 和 logs:CreateLogDelivery 权限。

单个日志组的数据保护策略所需的权限

Note

如果您在 Lambda 函数内执行任何此类操作，则 Lambda 执行角色和权限边界还必须包含以下权限。

操作	所需的 IAM 权限	资源
创建没有审计目标的数据保护策略	logs:PutDataProtectionPolicy	arn:aws:logs::log-group: <i>YOUR_LOG_GROUP</i> :*
将 CloudWatch Logs 作为审计目标来创建数据保护策略	logs:PutDataProtectionPolicy logs:CreateLogDelivery logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups	arn:aws:logs::log-group: <i>YOUR_LOG_GROUP</i> :* * * * *
将 Firehose 作为审计目标来创建数据保护策略	logs:PutDataProtectionPolicy logs:CreateLogDelivery firehose:TagDeliveryStream	arn:aws:logs::log-group: <i>YOUR_LOG_GROUP</i> :* * arn:aws:logs::deliverystream/ <i>YOUR_DELIVERY_STREAM</i>
将 Amazon S3 作为审计目标来创建数据保护策略	logs:PutDataProtectionPolicy logs:CreateLogDelivery s3:GetBucketPolicy	arn:aws:logs::log-group: <i>YOUR_LOG_GROUP</i> :* * arn:aws:s3::: <i>YOUR_BUCKET</i>

操作	所需的 IAM 权限	资源
	s3:PutBucketPolicy	arn:aws:s3::: <i>YOUR_BUCKET</i>
对已屏蔽的日志事件取消屏蔽	logs:Unmask	arn:aws:logs:::log-group: <i>YOUR_LOG_GROUP</i> :*
查看现有数据保护策略	logs:GetDataProtectionPolicy	arn:aws:logs:::log-group: <i>YOUR_LOG_GROUP</i> :*
删除数据保护策略	logs>DeleteDataProtectionPolicy	arn:aws:logs:::log-group: <i>YOUR_LOG_GROUP</i> :*

如果已经将任何数据保护审计日志发送到目标，则将日志发送到同一目标的其他策略只需要 logs:PutDataProtectionPolicy 和 logs:CreateLogDelivery 权限。

数据保护策略示例

以下示例策略允许用户创建、查看和删除数据保护策略，这些策略可以将审计调查发现发送到所有三种类型的审计目标。它不允许用户查看未屏蔽的数据。

JSON

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

```

        "Resource": "*"
    },
    {
        "Sid": "AllowDataProtectionAndBucketConfiguration",
        "Effect": "Allow",
        "Action": [
            "logs:GetDataProtectionPolicy",
            "logs:DeleteDataProtectionPolicy",
            "logs:PutDataProtectionPolicy",
            "s3:PutBucketPolicy",
            "firehose:TagDeliveryStream",
            "s3:GetBucketPolicy"
        ],
        "Resource": [
            "arn:aws:firehose:us-east-1:111122223333:deliverystream/delivery-  
stream-name",
            "arn:aws:s3:::amzn-s3-demo-destination-bucket",
            "arn:aws:logs:us-east-1:111122223333:log-group:log-group-name:*"
        ]
    }
]
}

```

创建账户范围的数据保护策略

您可以使用 CloudWatch Logs 控制台或 Amazon CLI 命令创建数据保护策略，以屏蔽账户中所有日志组中的敏感数据。这样做会同时影响当前的日志组和将来创建的日志组。

Important

将敏感数据摄入日志组时会进行检测并屏蔽。设置数据保护策略时，不会屏蔽在该时间之前摄入到日志组的日志事件。

主题

- [控制台](#)
- [Amazon CLI](#)

控制台

使用控制台创建账户范围数据保护策略

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Settings (设置)。它位于列表底部附近。
3. 选择日志选项卡。
4. 选择 配置。
5. 对于托管数据标识符，选择要为所有日志组审计和屏蔽的数据类型。您可以在选择框中键入以查找所需的标识符。

建议您仅选择与您的日志数据和业务相关的数据标识符。选择多种类型的数据可能会导致误报。

有关您可以保护的数据类型的详细信息，请参阅 [您可以保护的数据类型](#)。

6. (可选) 如果您想使用自定义数据标识符审核和屏蔽其他类型的数据，请选择添加自定义数据标识符。然后输入数据类型的名称和用于在日志事件中搜索该类型数据的正则表达式。有关更多信息，请参阅 [自定义数据标识符](#)。

单个数据保护策略最多可包含 10 个自定义数据标识符。定义自定义数据标识符的每个正则表达式不得超过 200 个字符。

7. (可选) 选择要将审核结果发送到的一项或多项服务。即使您选择不将审计结果发送到任何这些服务，您选择的敏感数据类型也仍将屏蔽。
8. 选择 Activate data protection (激活数据保护)。

Amazon CLI

使用 Amazon CLI 创建数据保护策略

1. 使用文本编辑器创建名为 DataProtectionPolicy.json 的策略文件。有关策略语法的信息，请参阅以下部分。
2. 输入以下命令：

```
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
```

Amazon CLI 或 API 操作的数据保护策略语法

当您创建用于 Amazon CLI 命令或 API 操作的 JSON 数据保护策略时，该策略必须包含两个 JSON 块：

- 第一个块必须包含 DataIdentifier 数组和带有 Audit 操作的 Operation 属性。DataIdentifier 数组列出要屏蔽的敏感数据类型。有关可用选项的更多信息，请参阅[您可以保护的数据类型](#)。

需要带有 Audit 操作的 Operation 属性才能找到敏感数据项。此 Audit 操作必须包含一个 FindingsDestination 对象。您可以选择使用该 FindingsDestination 对象列出要将审计结果报告发送到的一个或多个目标。如果指定日志组、Amazon Data Firehose 流和 S3 存储桶等目标，则它们必须已经存在。有关审计结果报告的示例，请参阅[审计结果报告](#)。

- 第二个块必须包含 DataIdentifier 数组和带有 Deidentify 操作的 Operation 属性。DataIdentifier 数组必须与策略第一个块中的 DataIdentifier 数组完全匹配。

带有 Deidentify 操作的 Operation 属性实际上是屏蔽数据的属性，它必须包含 "MaskConfig": {} 对象。"MaskConfig": {} 对象必须为空。

下面是仅使用托管数据标识符的数据保护策略示例。此策略屏蔽电子邮件地址和美国驾驶执照。

有关指定自定义数据标识符的策略信息，请参阅[在数据保护策略中使用自定义数据标识符](#)。

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

```

为单个日志组创建数据保护策略

您可以使用 CloudWatch Logs 控制台或 Amazon CLI 命令创建数据保护策略，以屏蔽敏感数据。

您可以为每个日志组分配一个数据保护策略。每个数据保护策略都可以对多种类型的信息进行审核。每个数据保护策略可以包含一条审核语句。

主题

- [控制台](#)
- [Amazon CLI](#)

控制台

使用控制台创建数据保护策略

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Logs (日志) 和 Log groups (日志组)。

3. 选择日志组的名称。
4. 选择 Actions (操作)、Create data protection policy (创建数据保护策略)。
5. 对于托管数据标识符，选择要在此日志组中审计和屏蔽的数据类型。您可以在选择框中键入以查找所需的标识符。

建议您仅选择与您的日志数据和业务相关的数据标识符。选择多种类型的数据可能会导致误报。

有关可以使用托管数据标识符保护哪些类型的数据的详细信息，请参阅 [您可以保护的数据类型](#)。

6. (可选) 如果您想使用自定义数据标识符审核和屏蔽其他类型的数据，请选择添加自定义数据标识符。然后输入数据类型的名称和用于在日志事件中搜索该类型数据的正则表达式。有关更多信息，请参阅 [自定义数据标识符](#)。

单个数据保护策略最多可包含 10 个自定义数据标识符。定义自定义数据标识符的每个正则表达式不得超过 200 个字符。

7. (可选) 选择要将审核结果发送到的一项或多项服务。即使您选择不将审计结果发送到任何这些服务，您选择的敏感数据类型也仍将屏蔽。
8. 选择 Activate data protection (激活数据保护)。

Amazon CLI

使用 Amazon CLI 创建数据保护策略

1. 使用文本编辑器创建名为 DataProtectionPolicy.json 的策略文件。有关策略语法的信息，请参阅以下部分。
2. 输入以下命令：

```
aws logs put-data-protection-policy --log-group-identifier "my-log-group" --policy-document file:///Path/DataProtectionPolicy.json --region us-west-2
```

Amazon CLI 或 API 操作的数据保护策略语法

当您创建用于 Amazon CLI 命令或 API 操作的 JSON 数据保护策略时，该策略必须包含两个 JSON 块：

- 第一个块必须包含 DataIdentifier 数组和带有 Audit 操作的 Operation 属性。DataIdentifier 数组列出要屏蔽的敏感数据类型。有关可用选项的更多信息，请参阅[您可以保护的数据类型](#)。

需要带有 Audit 操作的 Operation 属性才能找到敏感数据项。此 Audit 操作必须包含一个 FindingsDestination 对象。您可以选择使用该 FindingsDestination 对象列出要将审计结果报告发送到的一个或多个目标。如果指定日志组、Amazon Data Firehose 流和 S3 存储桶等目标，则它们必须已经存在。有关审计结果报告的示例，请参阅 [审计结果报告](#)。

- 第二个块必须包含 DataIdentifier 数组和带有 Deidentify 操作的 Operation 属性。DataIdentifier 数组必须与策略第一个块中的 DataIdentifier 数组完全匹配。

带有 Deidentify 操作的 Operation 属性实际上是屏蔽数据的属性，它必须包含 "MaskConfig": {} 对象。"MaskConfig": {} 对象必须为空。

以下是屏蔽电子邮件地址和美国驾驶执照的数据保护策略示例。

```
{
  "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": {}
      }
    }
  }
}
```

查看未屏蔽的数据

要查看未屏蔽的数据，用户必须具有 `logs:Unmask` 权限。具有此权限的用户可通过以下方式查看未屏蔽的数据：

- 查看日志流中的事件时，选择 **Display**（显示）、**Unmask**（取消屏蔽）。
- 使用包含 `unmask(@message)` 命令的 CloudWatch Logs Insights 查询。以下示例查询显示流中未屏蔽的 20 个最新日志事件：

```
fields @timestamp, @message, unmask(@message)
| sort @timestamp desc
| limit 20
```

有关 CloudWatch Logs Insights 命令的更多信息，请参阅 [CloudWatch Logs Insights 语法查询语法](#)。

- 使用带有 `unmask` 参数的 [GetLogEvents](#) 或 [FilterLogEvents](#) 操作。

`CloudWatchLogsFullAccess` 策略包含 `logs:Unmask` 权限。要向没有 `CloudWatchLogsFullAccess` 的用户授权 `logs:Unmask`，您可以向该用户附加自定义 IAM policy。有关更多信息，请参阅 [向用户添加权限（控制台）](#)。

审计结果报告

如果将 CloudWatch Logs 数据保护审计策略设置为将审计报告写入 CloudWatch Logs、Amazon S3 或 Firehose，则这些调查发现报告类似于以下示例。CloudWatch Logs 会为每个包含敏感数据的日志事件写入一份结果报告。

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

报告中的字段如下：

- `resourceArn` 字段显示发现敏感数据的日志组。
- `dataIdentifiers` 对象显示有关您正在审计的一种敏感数据的发现结果的信息。
- `name` 字段标识此部分报告的敏感数据类型。
- `count` 字段显示此类敏感数据在日志事件中出现的次数。
- `start` 和 `end` 字段按字符计数显示日志事件中每次出现敏感数据的位置。

前面的示例显示了在一个日志事件中找到两个电子邮件地址的报告。第一个电子邮件地址从日志事件的第 13 个字符开始，到第 26 个字符结束。第二个电子邮件地址从第 30 个字符到第 43 个字符。即使此日志事件有两个电子邮件地址，`LogEventsWithFindings` 指标的值也只递增一，因为该指标计算包含敏感数据的日志事件的数量，而不是敏感数据的出现次数。

所需的密钥政策用于将审计调查结果发现发送给 Amazon KMS 保护的存储桶

您可以通过启用 Amazon S3 托管式密钥的服务器端加密 (SSE-S3) 或 KMS 密钥的服务器端加密 (SSE-KMS) 来保护 Amazon S3 存储桶中的数据。有关详情，请参阅《Amazon S3 用户指南》中的[使用服务器端加密保护数据](#)。

如果将审计调查结果发送到 SSE-S3 保护的存储桶，则不需要额外的配置。Amazon S3 处理加密密钥。

如果将审计调查发现发送到 SSE-KMS 保护的存储桶，则您必须更新 KMS 密钥的密钥政策，以确保日志传输账户可以写入您的 S3 存储桶。有关与 SSE-KMS 结合使用时必需的密钥策略的更多信息，请参阅 Amazon CloudWatch Logs 用户指南中的 [Amazon S3](#)。

您可以保护的数据类型

本节包含有关可以在 CloudWatch Logs 数据保护策略中保护的数据类型的信息。CloudWatch Logs 托管数据标识符提供预配置的数据类型，用于保护财务数据、个人健康信息 (PHI) 和个人身份信息 (PII)。您还可以使用自定义数据标识符来创建针对特定使用场景量身定制的数据标识符。

目录

- [敏感数据类型的 CloudWatch Logs 托管数据标识符](#)
 - [凭证](#)
 - [凭证数据类型的数据标识符 ARN](#)
 - [设备标识符](#)
 - [设备数据类型的数据标识符 ARN](#)
 - [财务信息](#)
 - [财务数据类型的数据标识符 ARN](#)
 - [受保护健康信息 \(PHI \)](#)
 - [受保护健康信息 \(PHI, Protected Health Information\) 数据类型的数据标识符 ARN](#)
 - [个人信息 \(PII \)](#)
 - [驾驶执照识别号的关键字](#)
 - [国民身份证号的关键字](#)
 - [护照号码的关键字](#)
 - [纳税人识别号和参考号的关键字](#)
 - [个人信息 \(PII, Personally Identifiable Information\) 的数据标识符 ARN](#)

- [自定义数据标识符](#)
 - [什么是自定义数据标识符？](#)
 - [自定义数据标识符限制](#)
 - [在控制台中使用自定义数据标识符](#)
 - [在数据保护策略中使用自定义数据标识符](#)

敏感数据类型的 CloudWatch Logs 托管数据标识符

本节包含有关可使用托管数据标识符保护的数据类型的信息，以及哪些国家和地区与每种类型的数据相关。

对于某些类型的敏感数据，CloudWatch Logs 数据保护会扫描与数据接近的关键字，仅当找到该关键字时才会找到匹配项。如果关键字必须靠近特定类型的数据，则该关键字通常必须在 30 个字符以内（含）数据。

如果关键字包含空格，则 CloudWatch Logs 数据保护会自动匹配缺少空格的关键字变体，或包含下划线（_）或连字符（-）而不是空格的关键字变体。在某些情况下，CloudWatch Logs 还会扩展或缩写关键字以应对该关键字的常见变体。

下表列出 CloudWatch Logs 可以使用托管数据标识符检测的凭证、设备、财务、医疗和受保护健康信息（PHI）类型。除了特定类型的数据之外，这些数据可能也符合个人信息（PII, Personally Identifiable Information）的条件。

与语言和地区无关的受支持的标识符

标识符	类别
Address	个人
AwsSecretKey	凭证
CreditCardExpiration	财务
CreditCardNumber	财务
CreditCardSecurityCode	财务
EmailAddress	个人

标识符	类别
IpAddress	个人
LatLong	个人
Name	个人
OpenSshPrivateKey	凭证
PgpPrivateKey	凭证
PkcsPrivateKey	凭证
PuttyPrivateKey	凭证
VehicleIdentificationNumber	个人

区域相关数据标识符必须包含标识符名称，然后是连字符，最后是两个字母 (ISO 3166-1 alpha-2) 代码。例如 DriversLicense-US。

支持的标识符，必须包含两个字母的国家或地区代码

标识符	类别	国家/地区和语言
BankAccountNumber	财务	DE、ES、FR、GB、IT、US
CepCode	个人	BR
Cnpj	个人	BR
CpfCode	个人	BR
DriversLicense	个人	AT、AU、BE、 BG、CA、CY、 CZ、DE、DK、EE、ES、FI、 FR、GB、GR、 HR、HU、IE、IT、LT、LU、 LV、MT、NL、

标识符	类别	国家/地区和语言
		PL、PT、RO、SE、SI、SK、US
DrugEnforcementAgencyNumber	健康	美国
ElectoralRollNumber	个人	GB
HealthInsuranceCardNumber	健康	EU
HealthInsuranceClaimNumber	健康	美国
HealthInsuranceNumber	健康	FR
HealthcareProcedureCode	健康	美国
IndividualTaxIdentificationNumber	个人	美国
InseeCode	个人	FR
MedicareBeneficiaryNumber	健康	美国
NationalDrugCode	健康	美国
NationalIdentificationNumber	个人	DE、ES、IT
NationalInsuranceNumber	个人	GB
NationalProviderId	健康	美国
NhsNumber	健康	GB
NieNumber	个人	ES
NifNumber	个人	ES
PassportNumber	个人	CA、DE、ES、FR、GB、IT、US

标识符	类别	国家/地区和语言
PermanentResidenceNumber	个人	CA
PersonalHealthNumber	健康	CA
PhoneNumber	个人	BR、DE、ES、FR、GB、IT、US
PostalCode	个人	CA
RgNumber	个人	BR
SocialInsuranceNumber	个人	CA
Ssn	个人	ES、US
TaxId	个人	DE、ES、FR、GB
ZipCode	个人	美国

凭证

CloudWatch Logs 数据保护可以找到以下类型的凭证。

数据类型	数据标识符 ID	所需关键字	国家和地区
Amazon私有访问密钥	AwsSecretKey	aws_secret_access_key , credentials , secret access key, secret key, set-awscredential	全部
OpenSSH 私有密钥	OpenSSHPrivateKey	无	全部
PGP 私有密钥	PgpPrivateKey	无	全部

数据类型	数据标识符 ID	所需关键字	国家和地区
Pkcs 私有密钥	PkcsPrivateKey	无	全部
PuTTY 私有密钥	PuttyPrivateKey	无	全部

凭证数据类型的数据标识符 ARN

下表列出了您可以添加到数据保护策略中的数据标识符的 Amazon 资源名称 (ARN)。

凭证数据标识符 ARN

```
arn:aws:dataprotection::aws:data-identifier/AwsSecretKey
```

```
arn:aws:dataprotection::aws:data-identifier/OpenSshPrivateKey
```

```
arn:aws:dataprotection::aws:data-identifier/PgpPrivateKey
```

```
arn:aws:dataprotection::aws:data-identifier/PkcsPrivateKey
```

```
arn:aws:dataprotection::aws:data-identifier/PuttyPrivateKey
```

设备标识符

CloudWatch Logs 数据保护可以找到以下类型的设备标识符。

数据类型	数据标识符 ID	所需关键字	国家和地区
IP 地址	IpAddress	无	全部

设备数据类型的数据标识符 ARN

下表列出了您可以添加到数据保护策略中的数据标识符的 Amazon 资源名称 (ARN, Amazon Resource Name)。

设备数据标识符 ARN

arn:aws:dataprotection::aws:data-identifier/IpAddress

财务信息

CloudWatch Logs 数据保护可以找到以下类型的财务信息。

如果您已设置数据保护策略，则无论日志组位于什么地理位置，CloudWatch Logs 都会扫描您指定的数据标识符。此表中 Countries and regions (国家和地区) 列中的信息指定是否必须在数据标识符后附加两个字母的国家/地区代码，以检测这些国家和地区的相应关键字。

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
银行账户	BankAccountNumber	是。不同的关键字适用于不同的国家/地区。有关详细信息，请参阅本节后面的 Keywords for bank account numbers (银行账号的关键字) 表。	法国、德国、意大利、西班牙、英国、美国	包括由最多 34 个字母数字字符组成的国际银行账号 (I BANs) ，包括国家/地区代码等元素。
信用卡到期日期	CreditCardExpiration	exp d, exp m, exp y, expiration , expiry	全部	
信用卡号	CreditCardNumber	account number, american express,	全部	检测要求数据

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
		amex, bank card, card, card number, card num, cc #, ccn, check card, credit, credit card#, dankort, debit, debit card, diners club, discover, electron, japanese card bureau, jcb, mastercard , mc, pan, payment account number, payment card number, pcn, union pay, visa		为 13–19 位的数字序列，遵守 Luhn 验算公式，并为以下任意类型的信用卡使用标准卡号前缀：A merican Express、D ankort、Di ner’s Club、Disc over、Elec tron、Japa nese Card Bureau (JC B) 、Master card、银联和 Visa。

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
信用卡验证码	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	全部	

银行账号的关键字

使用以下关键字来获取银行账号。这包括由最多 34 个字母数字字符组成的国际银行账号（IBANs），包括国家/地区代码等元素。

Country	关键词
法国	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
德国	account code, account number, accountno# , accountnumber# , bankleitzahl , bban, customer account id, customer account number, customer bank account id, geheimzahl , iban, kartenummer , kontonummer , kreditkartenummer , sepa
意大利	account code, account number, accountno# , accountnumber# , bban, codice bancario, conto bancario, customer account id,

Country	关键词
	customer account number, customer bank account id, iban, numero di conto
西班牙	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
英国	account code, account number, accountno# , accountnumber# , bban, customer account ID, customer account number, customer bank account id, iban, sepa
美国	bank account, bank acct, checking account, checking acct, deposit account, deposit acct, savings account, savings acct, chequing account, chequing acct

CloudWatch Logs 不会报告出现的以下序列，信用卡发卡机构保留这些序列供公开测试。

```
1220000000000003, 2222405343248877, 2222990905257051, 2223007648726984,
2223577120017656,
30569309025904, 34343434343434, 3528000700000000, 3530111333300000, 3566002020360505,
36148900647913,
36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237,
401288888881881,
4111111111111111, 4222222222222, 4444333322221111, 4462030000000000, 4484070000000000,
49118300000000,
4917300800000000, 4917610000000000, 4917610000000000003, 5019717010103742,
5105105105105100,
5111010030175156, 5185540810000019, 5200828282828210, 5204230080000017,
5204740009900014, 5420923878724339,
5454545454545454, 5455330760000018, 5506900490000436, 5506900490000444,
5506900510000234, 5506920809243667,
5506922400634930, 5506927427317625, 5553042241984105, 5555553753048194,
555555555554444, 5610591081018250,
6011000990139424, 6011000400000000, 6011111111111117, 630490017740292441,
630495060000000000,
6331101999990016, 6759649826438453, 6799990100000000019, and 76009244561.
```

财务数据类型的数据标识符 ARN

下表列出了您可以添加到数据保护策略中的数据标识符的 Amazon 资源名称 (ARN)。

财务数据标识符 ARN
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

受保护健康信息 (PHI)

CloudWatch Logs 数据保护可以找到以下类型的受保护健康信息 (PHI) 。

如果您已设置数据保护策略，则无论日志组位于什么地理位置，CloudWatch Logs 都会扫描您指定的数据标识符。此表中 Countries and regions (国家和地区) 列中的信息指定是否必须在数据标识符后附加两个字母的国家/地区代码，以检测这些国家和地区的相应关键字。

数据类型	数据标识符 ID	所需关键字	国家和地区
毒品管理局 (DEA, Drug Enforcement Agency) 注册号	DrugEnforcementAgencyNumber	dea number, dea registration	美国

数据类型	数据标识符 ID	所需关键字	国家和地区
健康保险卡号 (EHIC)	HealthInsuranceCardNumber	assicurazione sanitaria numero, carta assicurazione numero, carte d'assuran ce maladie , carte européenne d'assuran ce maladie , ceam, ehic, ehic#, finlandeh icnumber# , gesundhei tskarte , hälsokort , health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherun gskarte , krankenve rsicherungsnummer , medical account number, numero conto medico, numéro d'assuran ce maladie , numéro de carte d'assuran ce , numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanho itokortin , sairausva kuutuskortti , sairausvakuutusnum ero , sjukförsäkring	欧盟

数据类型	数据标识符 ID	所需关键字	国家和地区
		nummer, sjukförsäkringskort , suomi ehic-numero , tarjeta de salud, terveysto rtti , tessera sanitaria assicuraz ione numero , versicher ungsnummer	
健康保险索赔编号 (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hcn, hcn#, hcnno#	美国
健康保险或医疗识别号	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	法国
医疗保健通用程序编码系统 (HCPCS) 代码	HealthcareProcedureCode	current procedural terminology , hcpcs, healthcare common procedure coding system	美国
医疗保险受益人号码 (MBN, Medicare Beneficiary Number)	MedicareBeneficiaryNumber	mbi, medicare beneficia ry	美国
国家药品编码 (NDC)	NationalDrugCode	national drug code, ndc	美国
国家提供商识别码 (NPI)	NationalProviderId	hipaa, n.p.i., national provider, npi	美国

数据类型	数据标识符 ID	所需关键字	国家和地区
国家健康服务 (NHS, National Health Service) 号码	NhsNumber	national health service, NHS	英国
个人健康号码	PersonalHealthNumber	canada healthcare number, msp number, care number, phn, soins de santé	加拿大

受保护健康信息 (PHI, Protected Health Information) 数据类型的数据标识符 ARN

下表列出了可以在受保护健康信息 (PHI) 数据保护策略中使用的数据标识符 Amazon 资源名称 (ARN) 。

PHI 数据标识符 ARN

```
arn:aws:dataprotection::aws:data-identifier/DrugEnforcementAgencyNumber-US
```

```
arn:aws:dataprotection::aws:data-identifier/HealthcareProcedureCode-US
```

```
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceCardNumber-EU
```

```
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceClaimNumber-US
```

```
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceNumber-FR
```

```
arn:aws:dataprotection::aws:data-identifier/MedicareBeneficiaryNumber-US
```

```
arn:aws:dataprotection::aws:data-identifier/NationalDrugCode-US
```

PHI 数据标识符 ARN
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

个人身份信息 (PII)

CloudWatch Logs 数据保护可以找到以下类型的个人身份信息 (PII)。

如果您已设置数据保护策略，则无论日志组位于什么地理位置，CloudWatch Logs 都会扫描您指定的数据标识符。此表中 Countries and regions (国家和地区) 列中的信息指定是否必须在数据标识符后附加两个字母的国家/地区代码，以检测这些国家和地区的相应关键字。

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
出生日期	DateOfBirth	dob, date of birth, birthdate , birth date, birthday, b-day, bday	任何	支持包括大多数日期格式，例如所有数字以及数字和月份名称的组合。日期组件可以用空格、斜

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
				杠 (/) 或连字符 (-) 分隔。
Código de Endereçamento Postal (CEP)	CepCode	cep, código de endereçamento postal, código de endereçamento postal	巴西	
Cadastro Nacional da Pessoa Jurídica (CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj	巴西	
Cadastro de Pessoas Físicas (CPF)	CpfCode	Cadastro de pessoas físicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf	巴西	

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
驾驶执照识别号	DriversLicense	是。不同的关键字适用于不同的国家/地区。有关详细信息，请参阅本节后面的 Drivers license identification numbers (驾驶执照识别号) 表。	许多国家/地区。有关详细信息，请参阅 Drivers license identification numbers 驾驶执照识别号) 表。	
选民名册编号	Electoral RollNumber	electoral #, electoral number, electoral roll #, electoral roll no., electoral roll number, electoral rollno	英国	
个人纳税人识别号	IndividualTaxIdentificationNumber	是。不同的关键字适用于不同的国家/地区。有关详细信息，请参阅本节后面的 Individual taxpayer identification numbers (个人纳税人识别号) 表。	巴西、法国、德国、西班牙、英国	

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
国家统计局与经济研究所 (INSEE, National Institute for Statistics and Economic Studies)	InseeCode	是。不同的关键字适用于不同的国家/地区。有关详细信息，请参阅本节后面的 Keywords for national identification numbers (国民身份证号的关键词) 表。	法国	
国民身份证号码	NationalIdentificationNumber	是。有关详细信息，请参阅本节后面的 Keywords for national identification numbers (国民身份证号的关键词) 表。	德国、意大利、西班牙	这包括 Documento Nacional de Identidad (DNI) 识别号 (西班牙)、Codice fiscale codes (意大利) 和国民身份证号 (德国)。
国民保险号码 (NINO)	NationalInsuranceNumber	insurance no., insurance number, insurance# , national insurance number, nationalinsurance# , nationalinsurancenumber , nin, nino	英国	–

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
Número de identidad de extranjero (NIE)	NieNumber	是。不同的关键字适用于不同的国家/地区。有关详细信息，请参阅本节后面的 Individual taxpayer identification numbers (个人纳税人识别号) 表。	西班牙	
Número de Identificación Fiscal (NIF)	NifNumber	是。不同的关键字适用于不同的国家/地区。有关详细信息，请参阅本节后面的 Individual taxpayer identification numbers (个人纳税人识别号) 表。	西班牙	
护照编号	PassportNumber	是。不同的关键字适用于不同的国家/地区。有关详细信息，请参阅本节后面的护照号码的关键字表。	加拿大、法国、德国、意大利、西班牙、英国、美国	

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
永久居留号码	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	加拿大	

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
电话号码	PhoneNumber	巴西：关键字还包括 ：cel、celular、fone、m residencial、numero residencial、telefone 其 他：cell、contact、fax、 number、mobile、phone、 number、tel、telephone 、telephone number	巴西、 加拿大、法国、德国、意大利、西班牙、英国、美国	这包括美国的免费电话号码和传真号码。如果关键字靠近数据，则数字不必包含国家/地区代码。如果关键字并不靠近数据，则数字必须包含国家/地区代码。
邮政编码	PostalCode	无	加拿大	
Registro Geral (RG)	RgNumber	是。不同的关键字适用于不同的国家/地区。有关详细信息，请参阅本节后面的 Individual taxpayer identification numbers (个人纳税人识别号) 表。	巴西	

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
社会保险号码 (SIN, Social Insurance Number)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	加拿大	
社会保障号码 (SSN)	Ssn	西班牙 - número de la seguridad social、social security no.、social security no、número de la seguridad social、social security number、socialsecurityno# 、ssn、ssn# 美国 - social security、ss#、ssn	西班牙、美国	
纳税人识别号或参考号	TaxId	是。不同的关键字适用于不同的国家/地区。有关详细信息，请参阅本节后面的 Individual taxpayer identification numbers (个人纳税人识别号) 表。 .	法国、德国、西班牙、英国	这包括 TIN (法国)；Steueridentifikationsnummer (德国)；CIF (西班牙)；以及 TRN、UTR (英国)。

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
邮政编码	ZipCode	zip code, zip+4	美国	美国邮政编码。
邮寄地址	Address	无	澳大利亚、加拿大、法国、德国、意大利、西班牙、英国、美国	尽管关键字并非必需，但检测过程要求地址包含城市或地点的名称以及邮政编码。
电子邮件地址	EmailAddress	无	任何	

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
全球定位系统 (GPS, Global Positioning System) 坐标	LatLong	coordinate , coordinates ,lat long, latitude longitude ,location, position	任何	CloudWatch Logs 可以检测纬度和经度坐标成对存储并且采用十进制度数 (DD, Decimal Degrees) 格式的 GPS 坐标，例如 41.948614 ,-87.6553 11。不支持采用度十进制分 (DDM, Degrees Decimal Minutes) 格式的坐标，例如 41°56.916 8'N 87°39.318

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
				7'W , 也不支持度分秒 (DMS, Degrees, Minutes, Seconds) 格式的坐标 , 例如 41°56'55.0104"N 87°39'19.1196"W。
全名	Name	无	任何	CloudWatch Logs 只能检测全名。只支持拉丁字符集。

数据类型	数据标识符 ID	所需关键字	国家和地区	备注
车辆识别号码 (VIN, Vehicle Identification Number)	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	任何	CloudWatch Logs 可以检测由 17 个字符序列组成并符合 ISO 3779 和 3780 标准的 VIN。这些标准旨在供全球使用。

驾驶执照识别号的关键字

为了检测各种类型的驾驶执照识别号，CloudWatch Logs 要求关键字的位置必须靠近数字。下表列出了 CloudWatch Logs 识别的特定国家和地区的关键字。

国家或地区	关键词
澳大利亚	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

国家或地区	关键词
奥地利	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik österreich
比利时	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
保加利亚	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
加拿大	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, permis de conduire
克罗地亚	vozačka dozvola
塞浦路斯	άδεια οδήγησης
捷克共和国	číslo licence, číslo licence řidiče, číslo řidičskéh o průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský průkaz, řidičský průkaz
丹麦	kørekort, kørekortnummer

国家或地区	关键词
爱沙尼亚	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
芬兰	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
法国	permis de conduire
德国	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
希腊	δεία οδήγησης, adeia odigisis
匈牙利	illesztőprogramok lic, jogosítvány, jogsí, licencszám, vezető engedély, vezetői engedély
爱尔兰	ceadúnas tiomána
意大利	patente di guida, patente di guida numero, patente guida, patente guida numero
拉脱维亚	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
立陶宛	vairuotojo pažymėjimas
卢森堡	fahrerlaubnis, führerschäin
马耳他	liċenzja tas-sewqan
荷兰	permis de conduire, rijbewijs, rijbewijsnummer

国家或地区	关键词
波兰	numer licencyjny, prawo jazdy, zezwolenie na prowadzenie
葡萄牙	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
罗马尼亚	numărul permisului de conducere, permis de conducere
斯洛伐克	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
斯洛文尼亚	vozniško dovoljenje
西班牙	carnet conducir, el carnet de conducir, licencia conducir, licencia de manejo, número carnet conducir, número de carnet de conducir, número de permiso conducir, número de permiso de conducir, número licencia conducir, número permiso conducir, permiso conducción, permiso conducir, permiso de conducción
瑞典	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.

国家或地区	关键词
英国	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
美国	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

国民身份证号的关键词

为了检测各种类型的国民身份证号，CloudWatch Logs 要求关键字的位置必须靠近数字。这包括 Documento Nacional de Identidad (DNI) 标识符（西班牙）、French National Institute for Statistics and Economic Studies（INSEE）代码、德国国民身份证号码，以及注册总署（RG, Registro Geral）号码（巴西）。

下表列出了 CloudWatch Logs 识别的特定国家和地区的关键字。

国家或地区	关键词
巴西	registro geral, rg
法国	assurance sociale, carte nationale d'identit é, cni, code sécurité sociale, French social security number, fssn#, insee, insurance number, national id number, nationalid#,

国家或地区	关键词
	numéro d'assurance, sécurité sociale, sécurité sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
德国	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
意大利	codice fiscale, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
西班牙	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#

护照号码的关键字

为了检测各种类型的护照号码，CloudWatch Logs 要求关键字的位置必须靠近数字。下表列出了 CloudWatch Logs 识别的特定国家和地区的关键字。

国家或地区	关键词
加拿大	passport, passport#, passport, passport#, passportno, passportno#
法国	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non

国家或地区	关键词
德国	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reiseepass, reiseepass-nr, reiseepassnummer
意大利	italian passport number, numéro passeport, numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
西班牙	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
英国	passeport #, passeport n °, passeportNon, passeport non, passeportn °, passport #, passport no, passport number, passport#, passportid
美国	passport, travel document

纳税人识别号和参考号的关键字

为了检测各种类型的纳税人识别号和参考号，CloudWatch Logs 要求关键字的位置必须靠近数字。下表列出了 CloudWatch Logs 识别的特定国家和地区的关键字。

国家或地区	关键词
巴西	cadastro de pessoa física, cadastro de pessoa física, cadastro de pessoas físicas, cadastro de pessoas físicas, cadastro nacional da pessoa jurídica, cadastro nacional da pessoa jurídica, cnpj, cpf
法国	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#

国家或地区	关键词
德国	identifikationsnummer, steuer id, steueridentifikationsnummer, steuernummer, tax id, tax identification number, tax number
西班牙	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#
英国	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
美国	个人纳税人识别号 (itin, i.t.i.n.)

个人身份信息 (PII, Personally Identifiable Information) 的数据标识符 ARN

下表列出了您可以添加到数据保护策略中的个人身份信息 (PII) 数据标识符的 Amazon 资源名称 (ARN) 。

PII 数据标识符 ARN

```
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
```

PII 数据标识符 ARN

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

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

PII 数据标识符 ARN

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

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

PII 数据标识符 ARN

```
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
```

```
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
```

PII 数据标识符 ARN

```
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
```

自定义数据标识符

主题

- [什么是自定义数据标识符？](#)
- [自定义数据标识符限制](#)
- [在控制台中使用自定义数据标识符](#)
- [在数据保护策略中使用自定义数据标识符](#)

什么是自定义数据标识符？

自定义数据标识符 (CDI) 可让您定义自己的自定义正则表达式，这些正则表达式可以在您的数据保护策略中使用。使用自定义数据标识符，您可以指向[托管式数据标识符](#)无法提供的特定于业务的个人信息 (PII) 用例。例如，您可以使用自定义数据标识符来查找公司特定的员工 ID。自定义数据标识符可以与托管式数据标识符结合使用。

自定义数据标识符限制

CloudWatch Logs 自定义数据标识符具有以下限制：

- 每项数据保护策略最多支持 10 个自定义数据标识符。
- 自定义数据标识符名称最多可包含 128 个字符。支持以下字符：
 - 字母数字：(a-zA-Z0-9)
 - 符号：(' _ | ' ' ')
- RegEx 的最大长度为 200 个字符。支持以下字符：
 - 字母数字：(a-zA-Z0-9)
 - 符号：(' _ | # | = | ' @ | / | ; | ' | ' | ' ' ')
 - RegEx 保留字符：(^ | \$ | ? | [|] | { | } | | | \ | * | + | ' | ')
- 自定义数据标识符不能与托管式数据标识符同名。
- 可以在账户级数据保护策略或日志组级数据保护策略中指定自定义数据标识符。与托管数据标识符类似，账户级策略中定义的自定义数据标识符与日志组级策略中定义的自定义数据标识符结合使用。

在控制台中使用自定义数据标识符

当您使用 CloudWatch 控制台创建或编辑数据保护策略时，要指定自定义数据标识符，只需输入数据标识符的名称和正则表达式。例如，您可以输入 **Employee_ID** 作为名称，并输入 **EmployeeID-\d{9}** 作为正则表达式。此正则表达式将检测并屏蔽 EmployeeID- 后面带有 9 个数字的日志事件。例如，EmployeeID-123456789

在数据保护策略中使用自定义数据标识符

如果使用 Amazon CLI 或 Amazon API 指定自定义数据标识符，则需要在用于定义数据保护策略的 JSON 策略中包含数据标识符名称和正则表达式。以下数据保护策略可检测并屏蔽带有公司特定员工 ID 的日志事件。

1. 在您的数据保护策略中创建一个 Configuration 块。
2. 为您的自定义数据标识符输入 Name。例如 **EmployeeId**。
3. 为您的自定义数据标识符输入 Regex。例如 **EmployeeID-\d{9}**。此正则表达式将匹配包含 EmployeeID- 且 EmployeeID- 后面有 9 位数字的日志事件。例如，EmployeeID-123456789
4. 请参阅策略声明中的以下自定义数据标识符。

```

{
  "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. (可选) 根据需继续向 Configuration 块添加其他自定义数据标识符。数据保护策略目前支持最多 10 个自定义数据标识符。

在摄取期间转换日志

通过日志转换和丰富，您可以在将日志摄取到 CloudWatch Logs 时以一致且上下文丰富的格式规范化所有日志。您可以使用预配置的模板为常见的 Amazon 服务（例如 Amazon WAF 和 Amazon Route 53）添加日志结构，或者使用 [Grok](#) 等原生解析器构建自定义转换器。您还可以重命名现有属性，并向日志添加其他元数据（例如，账户 ID 和区域）。

日志转换有助于简化和缩短应用程序中的日志查询，并有助于简化在日志上创建警报的过程。此功能为常见日志类型提供转换，并为主要 Amazon 日志源（如 VPC 流日志、Route 53 和 Amazon RDS for PostgreSQL）提供开箱即用的转换模板。您可以使用预配置的转换模板，也可以创建自定义转换器以满足您的需求。

日志转换可帮助您管理从格式和属性名称差异很大的源发出的日志。

创建转换器后，摄取的日志事件将以标准格式转换和存储。您可以利用这些转换后的日志，通过以下功能加速您的分析体验：

- [字段索引](#)
- [CloudWatch Logs Insights 发现的字段](#)
- 使用 [指标筛选条件](#) 灵活地设置警报
- 通过 [订阅筛选条件](#) 进行转发
- 使用 [Contributor Insights](#) 从日志事件创建指标数据，您可以选择让 Contributor Insights 规则在转换日志事件之前或之后对其进行评估。

转换仅在日志摄取期间进行。您无法转换已摄取的日志事件。转换操作不可逆。原始日志和转换后的日志都存储在 CloudWatch Logs 中，并采用相同的保留策略。日志转换和丰富功能包含在现有的标准日志类摄取价格中。日志存储成本将根据转换后的日志大小计算，这可能会超过原始日志量。

Important

日志事件转换后，您必须使用 CloudWatch Logs Insights 查询来查看转换后的日志版本。 [GetLogEvents](#) 和 [FilterLogEvents](#) 操作仅返回日志事件的原始版本，即转换之前的版本。

除了转换为不同的格式外，您还可以使用其他上下文（例如，账户 ID、区域和关键字）丰富日志。这些信息是从日志组名称和静态关键字中提取的。

日志转换可帮助您处理从格式和属性名称差异很大的源发出的日志。

仅标准日志类中的日志组支持日志转换和丰富功能。

您可以为单个日志组创建转换器，也可以创建适用于账户中所有或多个日志组的账户级转换器。如果某个日志组具有日志组级转换器，则该转换器会覆盖原本适用于该日志组的任何账户级转换器。

主题

- [创建和管理日志转换器](#)
- [您可以使用的处理器](#)
- [转换指标和错误](#)

创建和管理日志转换器

日志转换器包括一个或多个一起处于逻辑管线中的处理器。每个处理器都按照它们在转换器配置中列出的顺序依次应用于日志事件。

有些处理器属于解析器类型。每个转换器必须至少有一个解析器，并且转换器中的第一个处理器必须是解析器。

有些解析器是针对特定类型的 Amazon 公开发布的日志进行了配置的内置解析器。

其他处理器类型包括字符串变异器、JSON 变异器和数据处理器。

您可以为单个日志组创建转换器，也可以创建适用于账户中所有或多个日志组的账户级转换器。如果某个日志组具有日志组级转换器，则该转换器会覆盖原本适用于该日志组的任何账户级转换器。在您的账户中，一个区域最多可以有 20 个账户级转换器。

在创建转换器时，必须遵循以下准则：

- 如果您包含针对某种 Amazon 公开发布的日志类型预配置的解析器，则它必须是转换器中列出的第一个处理器。只能在转换器中包含一个此类处理器。
- 只能在转换器中包含一个 grok 处理器。
- 转换器必须至少包含一个解析器类型的处理器。最多可以包含五个解析器类型的处理器。此五个限制包括内置解析器和可配置的解析器。
- 转换器最多可包含 20 个处理器。
- 只能在转换器中包含一个 addKeys 处理器。

- 只能在转换器中包含一个 copyValue 处理器。
- 每个转换器最多可以从日志事件中提取 200 个字段。

有关所有受支持的处理器及其语法的更多信息，请参阅[您可以使用的处理器](#)。

主题

- [创建账户级转换器策略](#)
- [编辑或删除账户级转换器策略](#)
- [从头开始创建日志组级日志转换器](#)
- [通过复制现有转换器来创建日志组级转换器](#)
- [编辑日志组级转换器](#)
- [删除日志组级转换器](#)

创建账户级转换器策略

请按照本节中的步骤创建适用于账户中的所有日志组或者日志组名称以相同字符串（前缀）开头的多个日志组的转换器策略。一个区域中最多可以有 20 个账户级转换器策略。

您不能在同一区域创建两个使用相同前缀或一个前缀包含在另一个前缀中的转换器策略。例如，如果您为字符串前缀 /aws/lambda 创建了一个转换器策略，则不能再使用前缀 /aws 创建另一个转换器策略。但是，您可以为 /aws/lambda 创建一个转换器策略，而为 /aws/waf 创建另一个转换器策略。

创建账户级转换器策略

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在左侧导航窗格上，选择设置，然后选择日志选项卡。
3. 在账户的转换器策略部分，选择创建转换器策略。
4. 对于转换器策略名称，输入新策略的名称。
5. 对于选择日志组，请执行下列操作之一：
 - 选择所有标准日志组，将转换器策略应用于账户中的所有标准类日志组。
 - 选择按前缀匹配选择日志组，将策略应用于名称以相同字符串开头的日志组子集。然后，在选择标准中输入这些日志组的前缀。
6. 在选择解析器区域中，使用解析器选择要包含在转换器中的解析器。

如果是针对某种 Amazon 公开发布的日志类型预配置的解析器，则无需为其指定任何配置。

如果是其他解析器，则需要指定其配置。有关更多信息，请参阅[可配置的解析器类型处理器](#)中该处理器的相关信息。

7. 要添加其他处理器，请选择选择处理器。然后在处理器框中选择所需的处理器，并填写配置参数。有关配置参数的信息，请参阅[您可以使用的处理器](#)中该处理器的相关部分。

请记住，处理器会按照您将日志事件添加到转换器的顺序对其进行操作。

8. （可选）要添加其他处理器，请选择 + 处理器并重复上一步。
9. （可选）您可以随时测试迄今为止您根据示例日志事件构建的转换器。为此，请在转换器预览部分执行以下操作之一：
 - 在选择日志组中最多选择五个日志组，然后选择加载最新的日志事件。然后选择测试转换器。
 - 将日志事件直接复制到示例日志事件中，然后选择测试转换器。

随后会显示日志的转换后版本。

10. 添加完处理器并对示例日志的测试感到满意后，请选择保存。
11. 完成后，选择 Create (创建)。

编辑或删除账户级转换器策略

请按照本部分中的步骤编辑或删除账户级转换器策略。

编辑或删除账户级转换器策略

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在左侧导航窗格上，选择设置，然后选择日志选项卡。
3. 在转换器账户策略部分中，选择管理。
4. 选择要管理的转换器策略旁边的按钮，然后选择编辑或删除。

如果您正在编辑策略，请参阅[可配置的解析器类型处理器](#)中的步骤 5-11 以查看您的选项。

从头开始创建日志组级日志转换器

请按照以下步骤从头开始创建日志组级转换器。

使用控制台为日志组创建日志转换器

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。

2. 在导航窗格中，依次选择 Logs (日志) 和 Log groups (日志组)。
3. 选择要为其创建转换器的日志组。
4. 选择转换器选项卡。您可能需要向右滚动选项卡列表才能看到它。
5. 选择创建转换器。
6. 在选择解析器框中，选择要包含在转换器中的解析器。

如果是针对某种 Amazon 公开发布的日志类型预配置的解析器，则无需为其指定任何配置。

如果是其他解析器，则需要指定其配置。有关更多信息，请参阅[可配置的解析器类型处理器](#)中该处理器的相关信息。

7. 要添加其他处理器，请选择 + 添加处理器。然后在选择处理器框中选择所需的处理器，并填写配置参数。有关配置参数的信息，请参阅[您可以使用的处理器](#)中该处理器的相关部分。

请记住，处理器会按照您将日志事件添加到转换器的顺序对其进行操作。

8. (可选) 您可以随时测试迄今为止您根据示例日志事件构建的转换器。为此，请执行以下操作：
 - 在转换预览部分，选择加载示例日志以从此转换器适用的日志组加载示例日志事件，或者将日志事件粘贴到文本框中。

选择测试转换器。随后会显示日志的转换后版本

9. 添加完处理器并对示例日志的测试感到满意后，请选择保存。

使用 Amazon CLI 从头开始创建日志转换器

- 使用 `aws logs put-transformer` 命令。当使用 `parseJSON` 作为第一个处理器时，您必须使用 `@message` 作为源字段来解析整个日志事件。在初始 JSON 解析后，您可以在后续处理器中操作特定字段。以下示例创建了一个包含 `parseJSON` 和 `addKeys` 处理器的转换器：

```
aws logs put-transformer \  
  --transformer-config '[{"parseJSON":{"source":"@message"}}, {"addKeys":  
{"entries":[{"key":"metadata.transformed_in", "value":"CloudWatchLogs"},  
{"key":"feature", "value":"Transformation"}]}], {"trimString":{"withKeys":  
["status"]}]}' \  
  --log-group-identifier my-log-group-name
```

通过复制现有转换器来创建日志组级转换器

您可以使用控制台复制现有转换器的 JSON 配置。然后，您可以使用该代码通过 Amazon CLI 创建相同的转换器，也可以先修改配置。

通过复制现有日志转换器来创建日志转换器

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Logs (日志) 和 Log groups (日志组) 。
3. 选择包含要复制的转换器的日志组。
4. 选择转换选项卡。您可能需要向右滚动选项卡列表才能看到它。
5. 选择管理转换器。
6. 选择复制转换器。此时会将转换器 JSON 复制到剪贴板。
7. 创建一个文件并粘贴转换器配置。在此示例中，我们将调用文件 CopiedTransformer.json
8. 使用 Amazon CLI 创建具有该配置的新转换器。

```
aws logs put-transformer --log-group-identifier my-log-group-name \  
--transformer-config file://CopiedTransformer.json
```

编辑日志组级转换器

请按照以下步骤编辑现有的日志转换器。

编辑日志转换器

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，依次选择 Logs (日志) 和 Log groups (日志组) 。
3. 选择包含要编辑的转换器的日志组。
4. 选择转换选项卡。您可能需要向右滚动选项卡列表才能看到它。
5. 选择管理转换器。
6. 在解析器和处理器部分，进行更改。
7. 要添加其他处理器，请选择 + 添加处理器。然后在处理器框中选择所需的处理器，并填写配置参数。有关配置参数的信息，请参阅[您可以使用的处理器](#)中该处理器的相关部分。

请记住，处理器会按照您将日志事件添加到转换器的顺序对其进行操作。

8. (可选) 您可以随时测试迄今为止您根据示例日志事件构建的转换器。为此, 请执行以下操作:
 - 在转换预览部分, 选择加载示例日志以从此转换器适用的日志组加载示例日志事件, 或者将日志事件粘贴到文本框中。

选择测试转换。随后会显示日志的转换后版本

9. 添加完处理器并对示例日志的测试感到满意后, 请选择保存。

删除日志组级转换器

请按照以下步骤删除日志转换器。

删除日志转换器

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中, 依次选择 Logs (日志) 和 Log groups (日志组)。
3. 选择包含要编辑的转换器的日志组。
4. 选择转换选项卡。您可能需要向右滚动选项卡列表才能看到它。
5. 选择删除。
6. 在确认框中, 选择删除策略。

您可以使用的处理器

本节包含有关可以在日志事件转换器中使用的每个处理器的信息。处理器可分为解析器、字符串变异器、JSON 变异器和日期处理器。

目录

- [可配置的解析器类型处理器](#)
 - [parseJSON](#)
 - [grok](#)
 - [Grok 示例](#)
 - [示例 1：使用 grok 从非结构化日志中提取字段](#)
 - [示例 2：搭配使用 grok 和 parseJSON 从 JSON 日志事件中提取字段](#)
 - [示例 3：FIELD_NAME 中带有注释的 Grok 模式](#)

- [支持的 Grok 模式](#)
 - [常见日志格式示例](#)
 - [Apache 日志示例](#)
 - [NGINX 日志示例](#)
 - [Syslog 协议 \(RFC 5424\) 日志示例](#)
- [csv](#)
- [parseKeyValue](#)
- [用于 Amazon 公开发布的日志的内置处理器](#)
 - [parseWAF](#)
 - [parsePostgres](#)
 - [parseCloudfront](#)
 - [parseRoute53](#)
 - [parseVPC](#)
 - [parseToOCSF](#)
- [字符串变异处理器](#)
 - [lowerCaseString](#)
 - [upperCaseString](#)
 - [splitString](#)
 - [substituteString](#)
 - [trimString](#)
- [JSON 变异处理器](#)
 - [addKeys](#)
 - [deleteKeys](#)
 - [moveKeys](#)
 - [renameKeys](#)
 - [copyValue](#)
 - [listToMap](#)
- [数据类型转换器处理器](#)
 - [typeConverter](#)
- [您可以使用的处理器](#)
 - [datetimeConverter](#)

可配置的解析器类型处理器

parseJSON

parseJSON 处理器解析 JSON 日志事件，并将提取的 JSON 键值对插入到目标位置。如果没有指定目标位置，则处理器会将键值对放置在根节点下。当使用 parseJSON 作为第一个处理器时，您必须使用 @message 作为源字段来解析整个日志事件。在初始 JSON 解析后，您可以在后续处理器中操作特定字段。

原始 @message 内容不变，新键会添加到消息中。

字段	描述	必填？	默认值	限制
源	日志事件中待解析字段的路径。 请使用点表示法访问子字段。例如，store.book	否	@message	最大长度：128 最大嵌套键深度：3
目的地	已解析 JSON 的目标字段	否	Parent JSON node	最大长度：128 最大嵌套键深度：3

示例：

假设摄取的日志事件如下所示：

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

如果我们有此 parseJSON 处理器：

```
[
  {
    "parseJSON": {
      "destination": "new_key"
    }
  }
]
```

```
]

```

转换后的日志事件将如下所示。

```
{
  "new_key": {
    "outer_key": {
      "inner_key": "inner_value"
    }
  }
}
```

grok

使用 grok 处理器通过模式匹配来解析和构造非结构化数据。此处理器还可以从日志消息中提取字段。

字段	描述	必填？	默认值	限制	备注
源	要应用 Grok 匹配的字段路径	否	@message	最大长度：128 最大嵌套键深度：3	
match	要与日志事件匹配的 grok 模式	是		最大长度：512 最大 grok 模式：20 某些 grok 模式类型有单独的使用限制。以下模式可以任意组合使用，最多可使用五次：{URI, URIPARAM, URIPATHPARAM, SPACE, DATA, GREEDYDATA, GREEDYDATA_MULTILINE} Grok 模式不支持类型转换。	查看所有支持的 Grok 模式

字段	描述	必填？	默认值	限制	备注
				对于常见的日志格式模式 (APACHE_ACCESS_LOG , NGINX_ACCESS_LOG, SYSLOG5424) , 仅支持在常见的日志模式之后包含 DATA、GREEDYDATA 或 GREEDYDATA_MULTILINE 模式。	

Grok 模式的结构

下面是支持的 grok 模式结构：

```
%{PATTERN_NAME:FIELD_NAME}
```

- PATTERN_NAME：指用于匹配特定类型数据的预定义正则表达式。仅支持[受支持 grok 模式列表](#)中的预定义 grok 模式。不允许创建自定义模式。
- FIELD_NAME：为提取的值分配名称。FIELD_NAME 是可选的，但是如果您不指定此值，则提取的数据将从转换后的日志事件中删除。如果 FIELD_NAME 使用点表示法（例如“parent.child”），则将其视为 JSON 路径。
- 类型转换：不支持显式类型转换。请使用 [TypeConverter 处理器](#) 转换 grok 提取的任何值的数据类型。

要创建更复杂的匹配表达式，您可以组合多个 grok 模式。最多可以组合 20 个 grok 模式来匹配一个日志事件。例如，可以使用此模式组合 %{NUMBER:timestamp} [%{NUMBER:db} %{IP:client_ip}:%{NUMBER:client_port}] %{GREEDYDATA:data} 从 Redis 慢日志条目中提取字段，如下所示：

```
1629860738.123456 [0 127.0.0.1:6379] "SET" "key1" "value1"
```

Grok 示例

示例 1：使用 grok 从非结构化日志中提取字段

日志示例：

```
293750 server-01.internal-network.local OK "[Thread-000] token generated"
```

使用的转换器：

```
[
  {
    "grok": {
      "match": "%{NUMBER:version} %{HOSTNAME:hostname} %{NOTSPACE:status}
%{QUOTEDSTRING:logMsg}"
    }
  }
]
```

输出：

```
{
  "version": "293750",
  "hostname": "server-01.internal-network.local",
  "status": "OK",
  "logMsg": "[Thread-000] token generated"
}
```

日志示例：

```
23/Nov/2024:10:25:15 -0900 172.16.0.1 200
```

使用的转换器：

```
[
  {
    "grok": {
      "match": "%{HTTPDATE:timestamp} %{IPORHOST:clientip}
%{NUMBER:response_status}"
    }
  }
]
```

```
}  
]
```

输出：

```
{  
  "timestamp": "23/Nov/2024:10:25:15 -0900",  
  "clientip": "172.16.0.1",  
  "response_status": "200"  
}
```

示例 2：搭配使用 grok 和 parseJSON 从 JSON 日志事件中提取字段

日志示例：

```
{  
  "timestamp": "2024-11-23T16:03:12Z",  
  "level": "ERROR",  
  "logMsg": "GET /page.html HTTP/1.1"  
}
```

使用的转换器：

```
[  
  {  
    "parseJSON": {}  
  },  
  {  
    "grok": {  
      "source": "logMsg",  
      "match": "%{WORD:http_method} %{NOTSPACE:request} HTTP/  
%{NUMBER:http_version}"  
    }  
  }  
]
```

输出：

```
{  
  "timestamp": "2024-11-23T16:03:12Z",  
  "level": "ERROR",  
  "logMsg": "GET /page.html HTTP/1.1",  
}
```

```
"http_method": "GET",
"request": "/page.html",
"http_version": "1.1"
}
```

示例 3：FIELD_NAME 中带有点注释的 Grok 模式

日志示例：

```
192.168.1.1 GET /index.html?param=value 200 1234
```

使用的转换器：

```
[
  {
    "grok": {
      "match": "%{IP:client.ip} %{WORD:method} %{URIPATHPARAM:request.uri}
%{NUMBER:response.status} %{NUMBER:response.bytes}"
    }
  }
]
```

输出：

```
{
  "client": {
    "ip": "192.168.1.1"
  },
  "method": "GET",
  "request": {
    "uri": "/index.html?param=value"
  },
  "response": {
    "status": "200",
    "bytes": "1234"
  }
}
```

支持的 Grok 模式

下表列出了 grok 处理器支持的模式。

常规 grok 模式

Grok 模式	描述	最大模式限制	示例
USERNAME 或 USER	匹配一个或多个字符，可以包括小写字母 (a-z)、大写字母 (A-Z)、数字 (0-9)、点 (.)、下划线 (_) 或连字符 (-)。	20	输入：user123.name-TEST 模式：%{USERNAME:name} 输出：{"name": "user123.name-TEST"}
INT	匹配可选的加号或减号，后跟一个或多个数字。	20	输入：-456 模式：%{INT:num} 输出：{"num": "-456"}
BASE10NUM	匹配带有可选符号和小数点的整数或浮点数	20	输入：-0.67 模式：%{BASE10NUM:num} 输出：{"num": "-0.67"}
BASE16NUM	匹配带有可选符号 (+ 或 -) 和可选 0x 前缀的十进制和十六进制数字	20	输入：+0xA1B2 模式：%{BASE16NUM:num} 输出：{"num": "+0xA1B2"}
POSINT	匹配不带前导零的正整数，由一个或多个数字组成 (1-9 后跟 0-9)	20	输入：123 模式：%{POSINT:num} 输出：{"num": "123"}
NONNEGINT	匹配任何整数 (由一个或多个数字 0-9 组成)，包括零和带前导零的数字。	20	输入：007 模式：%{NONNEGINT:num} 输出：{"num": "007"}

Grok 模式	描述	最大模式限制	示例
WORD	匹配由一个或多个单词字符 (\w) 组成的完整单词，包括字母、数字和下划线	20	输入：user_123 模式：%{WORD:user} 输出：{"user": "user_123"}
NOTSPACE	匹配一个或多个非空格字符。	5	输入：hello_world123 模式：%{NOTSPACE:msg} 输出：{"msg": "hello_world123"}
SPACE	匹配零个或多个空格字符。	5	输入：" " 模式：%{SPACE:extra} 输出：{"extra": " "}
DATA	匹配任意字符（换行符除外）零次或多次，不贪婪。	5	输入：abc def ghi 模式：%{DATA:x} %{DATA:y} 输出：{"x": "abc", "y": "def ghi"}
GREEDYDATA	匹配任意字符（换行符除外）零次或多次，贪婪。	5	输入：abc def ghi 模式：%{GREEDYDATA:x} %{GREEDYDATA:y} 输出：{"x": "abc def", "y": "ghi"}

Grok 模式	描述	最大模式限制	示例
GREEDYDATA_MULTILINE	匹配任意字符（包括换行符）零次或多次，贪婪。	1	输入： abc def ghi 模式：%{GREEDYDATA_MULTILINE:data} 输出：{"data": "abc\ndef\nghi"}
QUOTEDSTRING	将带引号的字符串（单引号或双引号）与转义字符匹配。	20	输入："Hello, world!" 模式：%{QUOTEDSTRING:msg} 输出：{"msg": "Hello, world!"}
UUID	匹配标准 UUID 格式：8 个十六进制字符，后跟三组 4 个十六进制字符，以 12 个十六进制字符结尾，全部用连字符分隔。	20	输入：550e8400-e29b-41d4-a716-446655440000 模式：%{UUID:id} 输出：{"id": "550e8400-e29b-41d4-a716-446655440000"}

Grok 模式	描述	最大模式限制	示例
URN	匹配 URN (统一资源名称) 语法。	20	输入 : urn:isbn: 0451450523 模式 : %{URN:urn} 输出 : {"urn": "urn:isbn: :0451450523"}

Amazon grok 模式

Pattern	描述	最大模式限制	示例
ARN	匹配 Amazon Amazon 资源名称 (ARN), 捕获分区 (aws、aws-cn 或 aws-us-gov)、服务、区域、账户 ID 以及最多 5 个以斜杠分隔的分层资源标识符。它不匹配冒号之间缺失信息的 ARN。	5	输入 : arn:aws:iam:us-east-1:123456789012:user/johndoe 模式 : %{ARN:arn} 输出 : {"arn": "arn:aws:iam:us-east-1:123456789012:user/johndoe"}

联网 grok 模式

Grok 模式	描述	最大模式限制	示例
CISCOMAC	匹配 4-4-4 十六进制格式的 MAC 地址。	20	输入 : 0123.4567.89AB 模式 : %{CISCOMAC:MacAddress} 输出 : {"MacAddress": "0123.4567.89AB"}

Grok 模式	描述	最大模式限制	示例
WINDOWSMA C	将十六进制格式的 MAC 地址与连字符匹配	20	输入：01-23-45-67-89-AB 模式：%{WINDOWS MAC:MacAddress} 输出：{"MacAddress": "01-23-45-67-89-AB"}
COMMONMAC	将十六进制格式的 MAC 地址与冒号匹配。	20	输入：01:23:45: 67:89:AB 模式：%{COMMONM AC:MacAddress} 输出：{"MacAddress": "01:23:45:67:89:AB"}
MAC	匹配 CISCOMAC、WINDOWSMA C 或 COMMONMAC grok 模式之一	20	输入：01:23:45: 67:89:AB 模式：%{MAC:m1} 输出：{"m1": "01 :23:45:67:89:AB"}
IPV6	匹配 IPv6 地址，包括压缩形式和 IPv4 映射的 IPv6 地址。	5	输入：2001:db8: 3333:4444:5555:666 6:7777:8888 模式：%{IPV6:ip} 输出：{"ip": "2001:db8 :3333:4444:5555:66 66:7777:8888"}

Grok 模式	描述	最大模式限制	示例
IPV4	匹配 IPv4 地址。	20	输入 : 192.168.0.1 模式 : %{IPV4:ip} 输出 : {"ip": "192.168.0.1"}
IP	匹配 %{IPv6} 支持的 IPv6 地址或 %{IPv4} 支持的 IPv4 地址	5	输入 : 192.168.0.1 模式 : %{IP:ip} 输出 : {"ip": "192.168.0.1"}
HOSTNAME 或 HOST	匹配域名，包括子域	5	输入 : server-01.internal-network.local 模式 : %{HOST:host} 输出 : {"host": "server-01.internal-network.local"}
IPORHOST	匹配主机名或 IP 地址	5	输入 : 2001:db8:3333:4444:5555:6666:7777:8888 模式 : %{IPORHOST:ip} 输出 : {"ip": "2001:db8:3333:4444:5555:6666:7777:8888"}

Grok 模式	描述	最大模式限制	示例
HOSTPORT	匹配 %{IPORHOST} 模式支持的 IP 地址或主机名，后跟冒号和端口号，并将端口号捕获为输出中的“PORT”。	5	输入：192.168.1.1:8080 模式：%{HOSTPORT:ip} 输出：{"ip":"192.168.1.1:8080","PORT":"8080"}
URIHOST	匹配 %{IPORHOST} 模式支持的 IP 地址或主机名，可后跟冒号和端口号，并将端口捕获为“port”（如果存在）。	5	输入：example.com:443 10.0.0.1 模式：%{URIHOST:host} %{URIHOST:ip} 输出：{"host":"example.com:443","port":"443","ip":"10.0.0.1"}

Path grok 模式

Grok 模式	描述	最大模式限制	示例
UNIXPATH	匹配 URL 路径，可能包含查询参数。	20	输入：/search?q=regex 模式：%{UNIXPATH:path} 输出：{"path":"/search?q=regex"}
WINPATH	匹配 Windows 文件路径。	5	输入：C:\Users\John\Documents\file.txt 模式：%{WINPATH:path} 输出：{"path": "C:\\Users\\John\\"

Grok 模式	描述	最大模式限制	示例
			Documents\\file.txt"} }
路径	匹配 URL 或 Windows 文件路径	5	输入：/search?q=regex 模式：%{PATH:path} 输出：{"path":"/search?q=regex"}
TTY	匹配终端和伪终端的 Unix 设备路径。	20	输入：/dev/tty1 模式：%{TTY:path} 输出：{"path":"/dev/tty1"}
URIPROTO	匹配字母，可后跟加号 (+) 字符和其他字母或加号 (+) 字符	20	输入：web+transformer 模式：%{URIPROTO:protocol} 输出：{"protocol":"web+transformer"}
URIPATH	匹配 URI 的路径组件	20	输入：/category/sub-category/product_name 模式：%{URIPATH:path} 输出：{"path":"/category/sub-category/product_name"}

Grok 模式	描述	最大模式限制	示例
URIPARAM	匹配 URL 查询参数	5	输入：<code>?param1=value1&param2=value2</code> 模式：<code>%{URIPARAM:url}</code> 输出：<code>{"url":"?param1=value1&param2=value2"}</code>
URIPATHPARAM	匹配 URI 路径，可后跟查询参数	5	输入：<code>/category/sub-category/product?id=12345&color=red</code> 模式：<code>%{URIPATHPARAM:path}</code> 输出：<code>{"path":"/category/sub-category/product?id=12345&color=red"}</code>
URI	匹配完整的 URI	5	输入：<code>https://user:password@example.com/path/to/resource?param1=value1&param2=value2</code> 模式：<code>%{URI:uri}</code> 输出：<code>{"path":"https://user:password@example.com/path/to/resource?param1=value1&param2=value2"}</code>

日期和时间 grok 模式

Grok 模式	描述	最大模式限制	示例
MONTH	将完整的或缩写的英文月份名称匹配为完整单词	20	输入：Jan 模式：%{MONTH:month} 输出：{"month": "Jan"} 输入：January 模式：%{MONTH:month} 输出：{"month": "January"}
MONTHNUM	匹配 1 到 12 之间的月份数字，个位数月份可选择使用前导零。	20	输入：5 模式：%{MONTHNUM:month} 输出：{"month": "5"} 输入：05 模式：%{MONTHNUM:month} 输出：{"month": "05"}
MONTHNUM	匹配 01 到 12 之间的两位数月份数字。	20	输入：05 模式：%{MONTHNUM2:month}

Grok 模式	描述	最大模式限制	示例
			输出：{"month": "05"}
MONTHDAY	匹配月份中的日期，从 1 到 31，可选择使用前导零。	20	输入：31 模式：%{MONTHDAY:monthDay} 输出：{"monthDay": "31"}
YEAR	匹配两位或四位数的年份	20	输入：2024 模式：%{YEAR:year} 输出：{"year": "2024"} 输入：24 模式：%{YEAR:year} 输出：{"year": "24"}
DAY	匹配完整或缩写的日期名称。	20	输入：Tuesday 模式：%{DAY:day} 输出：{"day": "Tuesday"}
HOURL	以 24 小时制格式匹配小时，可选择使用前导零 (0)0-23。	20	输入：22 模式：%{HOURL:hour} 输出：{"hour": "22"}

Grok 模式	描述	最大模式限制	示例
MINUTE	匹配分钟 (00-59)。	20	输入：59 模式：%{MINUTE:min} 输出：{"min":"59"}
SECOND	匹配代表秒 (0)0-60 的数字，可后跟小数点或冒号，以及一个或多个数字表示小数分钟	20	输入：3 模式：%{SECOND:second} 输出：{"second":"3"} 输入：30.5 模式：%{SECOND:minSec} 输出：{"minSec":"30.5"} 输入：30:5 模式：%{SECOND:minSec} 输出：{"minSec":"30:5"}
TIME	将时间格式与小时、分钟和秒匹配，格式为 (H)H:mm:(s)s。秒包括闰秒 (0)0-60。	20	输入：09:45:32 模式：%{TIME:time} 输出：{"time":"09:45:32"}

Grok 模式	描述	最大模式限制	示例
DATE_US	以 (M)M/(d)d/(yy)yy 或 (M)M-(d)d-(yy)yy 格式匹配日期。	20	输入：11/23/2024 模式：%{DATE_US:date} 输出：{"date": "11/23/2024"} 输入：1-01-24 模式：%{DATE_US:date} 输出：{"date": "1-01-24"}
DATE_EU	以 (d)d/(M)M/(yy)yy、(d)d-(M)M-(yy)yy 或 (d)d.(M)M.(yy)yy 格式匹配日期。	20	输入：23/11/2024 模式：%{DATE_EU:date} 输出：{"date": "23/11/2024"} 输入：1.01.24 模式：%{DATE_EU:date} 输出：{"date": "1.01.24"}

Grok 模式	描述	最大模式限制	示例
ISO8601_TIMEZONE	将 UTC 偏移量“Z”或时区偏移量与可选冒号进行匹配，格式为 [+-(H)H(:)mm。	20	输入：+05:30 模式：%{ISO8601_TIMEZONE:tz} 输出：{"tz":"+05:30"} 输入：-530 模式：%{ISO8601_TIMEZONE:tz} 输出：{"tz":"-530"} 输入：Z 模式：%{ISO8601_TIMEZONE:tz} 输出：{"tz":"Z"}
ISO8601_SECONDS	匹配代表秒 (0)0-60 的数字，可后跟小数点或冒号，以及一个或多个数字表示小数秒	20	输入：60 模式：%{ISO8601_SECONDS:second} 输出：{"second":"60"}

Grok 模式	描述	最大模式限制	示例
TIMESTAMP _ISO8601	匹配 ISO8601 日期时间格式 (yy)yy-(M)M-(d)dT(H)H:mm:((s)s)(Z +-(H)H:mm) 与可选秒数和时区。	20	输入：2023-05-15T14:30:00+05:30 模式：%{TIMESTAMP_ISO8601:timestamp} 输出：{"timestamp":"2023-05-15T14:30:00+05:30"} 输入：23-5-1T1:25+5:30 模式：%{TIMESTAMP_ISO8601:timestamp} 输出：{"timestamp":"23-5-1T1:25+5:30"} 输入：23-5-1T1:25Z 模式：%{TIMESTAMP_ISO8601:timestamp} 输出：{"timestamp":"23-5-1T1:25Z"}

Grok 模式	描述	最大模式限制	示例
DATE	使用 <code>%{DATE_US}</code> 匹配美国格式的日期，或者使用 <code>%{DATE_EU}</code> 匹配欧盟格式的日期	20	输入：11/29/2024 模式： <code>%{DATE:date}</code> 输出： <code>{"date":"11/29/2024"}</code> 输入：29.11.2024 模式： <code>%{DATE:date}</code> 输出： <code>{"date":"29.11.2024"}</code>
DATESTAMP	匹配 <code>%{DATE}</code> ，后跟 <code>%{TIME}</code> 模式，用空格或连字符分隔。	20	输入：29-11-2024 14:30:00 模式： <code>%{DATESTAMP:dateTime}</code> 输出： <code>{"dateTime":"29-11-2024 14:30:00"}</code>
TZ	匹配常见的时区缩写 (PST、PDT、MST、MDT、CST CDT、EST、EDT、UTC)。	20	输入：PDT 模式： <code>%{TZ:tz}</code> 输出： <code>{"tz":"PDT"}</code>

Grok 模式	描述	最大模式限制	示例
DATESTAMP_RFC822	以以下格式匹配日期和时间：Day MonthName (D)D (YY)YY (H)H:mm:(s)s Timezone	20	输入：Monday Jan 5 23 1:30:00 CDT 模式：%{DATESTAMP_RFC822:dateTime} 输出：{"dateTime":"Monday Jan 5 23 1:30:00 CDT"} 输入：Mon January 15 2023 14:30:00 PST 模式：%{DATESTAMP_RFC822:dateTime} 输出：{"dateTime":"Mon January 15 2023 14:30:00 PST"}

Grok 模式	描述	最大模式限制	示例
DATESTAMP_RFC2822	匹配 RFC2822 日期 时间格式 : Day, (d)d MonthName (yy)yy (H)H:mm:(s)s Z[+-(H) H:mm	20	输入 : Mon, 15 May 2023 14:30:00 +0530 模式 : % {DATESTA MP_RFC282 2:dateTime} 输出 : { "dateTim e": "Mon, 15 May 2023 14:30:00 +0530" } 输入 : Monday, 15 Jan 23 14:30:00 Z 模式 : % {DATESTA MP_RFC282 2:dateTime} 输出 : { "dateTim e": "Monday, 15 Jan 23 14:30:00 Z" }
DATESTAMP_OTHER	以以下格式匹配日期和时 间 : Day MonthName (d)d (H)H:mm:(s)s Timezone (yy)yy	20	输入 : Mon May 15 14:30:00 PST 2023 模式 : % {DATESTA MP_OTHER: dateTime} 输出 : { "dateTim e": "Mon May 15 14:30:00 PST 2023" }

Grok 模式	描述	最大模式限制	示例
DATESTAMP_EVENTLOG	匹配不带分隔符的紧凑日期时间格式：(yy)yyMM(d)d(H)Hmm(s)s	20	输入：20230515143000 模式：%{DATESTAMP_EVENTLOG:dateTime} 输出：{"dateTime":"20230515143000"}

日志 grok 模式

Grok 模式	描述	最大模式限制	示例
LOGLEVEL	匹配不同大小写和缩写形式的标准日志级别，包括：Alert/ALERT、Trace/TRACE、Debug/DEBUG、Notice/NOTICE、Info/INFO、Warn/Warning/WARN/WARNING、Err/Error/ERROR、Crit/Critical/CRITICAL、Fatal/FATAL、Severe/SEVERE、Emerg/Emergency/EMERG/EMERGENCY	20	输入：INFO 模式：%{LOGLEVEL:logLevel} 输出：{"logLevel":"INFO"}
HTTPDATE	匹配日志文件中经常使用的日期和时间格式。	20	输入：23/Nov/2024:14:30:00 +0640

Grok 模式	描述	最大模式限制	示例
	格式：(d)d/MonthName/(yy)yy:(H)H:mm:(s)s Timezone MonthName：匹配完整或缩写的英文月份名称（例如：“Jan”或“January”）时区：匹配 %{INT} grok 模式		模式：%{HTTPDATE:date} 输出：{"date":"23/Nov/2024:14:30:00+0640"}
SYSLOGTIMESTAMP	匹配日期格式 MonthName (d)d (H)H:mm:(s)s MonthName：匹配完整或缩写的英文月份名称（例如：“Jan”或“January”）	20	输入：Nov 29 14:30:00 模式：%{SYSLOGTIMESTAMP:dateTime} 输出：{"dateTime":"Nov 29 14:30:00"}
PROG	匹配由字母、数字、点、下划线、正斜杠、百分号和连字符组成的程序名称。	20	输入：user.profile/settings-page 模式：%{PROG:program} 输出：{"program":"user.profile/settings-page"}

Grok 模式	描述	最大模式限制	示例
SYSLOGPROG	匹配 PROG grok 模式，可后跟方括号中的进程 ID。	20	输入：user.profile/settings-page[1234] 模式：%{SYSLOGPROG:programWithId} 输出：{"programWithId": "user.profile/settings-page[1234]", "program": "user.profile/settings-page", "pid": "1234"}
SYSLOGHOST	匹配 %{HOST} 或 %{IP} 模式	5	输入：2001:db8:3333:4444:5555:6666:7777:8888 模式：%{SYSLOGHOST:ip} 输出：{"ip": "2001:db8:3333:4444:5555:6666:7777:8888"}

Grok 模式	描述	最大模式限制	示例
SYSLOGFACILITY	匹配十进制格式的 syslog 优先级。该值应括在尖括号 (<>) 中。	20	输入 : <13.6> 模式 : %{SYSLOGFACILITY:syslog} 输出 : {"syslog": "<13.6>", "facility": "13", "priority": "6"}

常见的 log grok 模式

您可以使用预定义的自定义 grok 模式来匹配 Apache、NGINX 和 Syslog 协议 (RFC 5424) 日志格式。使用这些特定模式时，它们必须是匹配配置中的第一个模式，并且不能有其他模式位于它们之前。此外，后面只有跟有一个 DATA。GREEDYDATA 或 GREEDYDATA_MULTILINE 模式。

Grok 模式	描述	最大模式限制
APACHE_ACCESS_LOG	匹配 Apache 访问日志	1
NGINX_ACCESS_LOG	匹配 NGINX 访问日志	1
SYSLOG5424	匹配 Syslog 协议 (RFC 5424) 日志	1

下面展示了使用这些常见日志格式模式的有效和无效示例。

```
"%{NGINX_ACCESS_LOG} %{DATA}" // Valid
"%{SYSLOG5424}%{DATA:logMsg}" // Valid
"%{APACHE_ACCESS_LOG} %{GREEDYDATA:logMsg}" // Valid
"%{APACHE_ACCESS_LOG} %{SYSLOG5424}" // Invalid (multiple common log patterns used)
```

```
"%{NGINX_ACCESS_LOG} %{NUMBER:num}" // Invalid (Only GREEDYDATA and DATA patterns are supported with common log patterns)
"%{GREEDYDATA:logMsg} %{SYSLOG5424}" // Invalid (GREEDYDATA and DATA patterns are supported only after common log patterns)
```

常见日志格式示例

Apache 日志示例

日志示例：

```
127.0.0.1 - - [03/Aug/2023:12:34:56 +0000] "GET /page.html HTTP/1.1" 200 1234
```

转换器：

```
[
  {
    "grok": {
      "match": "%{APACHE_ACCESS_LOG}"
    }
  }
]
```

输出：

```
{
  "request": "/page.html",
  "http_method": "GET",
  "status_code": 200,
  "http_version": "1.1",
  "response_size": 1234,
  "remote_host": "127.0.0.1",
  "timestamp": "2023-08-03T12:34:56Z"
}
```

NGINX 日志示例

日志示例：

```
192.168.1.100 - Foo [03/Aug/2023:12:34:56 +0000] "GET /account/login.html HTTP/1.1"
200 42 "https://www.amazon.com/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
```

转换器：

```
[
  {
    "grok": {
      "match": "%{NGINX_ACCESS_LOG}"
    }
  }
]
```

输出：

```
{
  "request": "/account/login.html",
  "referrer": "https://www.amazon.com/",
  "agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36",
  "http_method": "GET",
  "status_code": 200,
  "auth_user": "Foo",
  "http_version": "1.1",
  "response_size": 42,
  "remote_host": "192.168.1.100",
  "timestamp": "2023-08-03T12:34:56Z"
}
```

Syslog 协议 (RFC 5424) 日志示例

日志示例：

```
<165>1 2003-10-11T22:14:15.003Z mymachine.example.com evntslog - ID47
[exampleSDID@32473 iut="3" eventSource= "Application" eventID="1011"]
[examplePriority@32473 class="high"]
```

转换器：

```
[
  {
    "grok": {
      "match": "%{SYSLOG5424}"
    }
  }
]
```



```
}  
]
```

输出：

```
{  
  "pri": 165,  
  "version": 1,  
  "timestamp": "2003-10-11T22:14:15.003Z",  
  "hostname": "mymachine.example.com",  
  "app": "evntslog",  
  "msg_id": "ID47",  
  "structured_data": "exampleSDID@32473 iut=\"3\" eventSource= \"Application\" eventID=  
\"1011\"",  
  "message": "[examplePriority@32473 class=\"high\"]"  
}
```

CSV

csv 处理器将日志事件中的逗号分隔值 (CSV) 解析为列。

字段	描述	必填？	默认值	限制
源	日志事件中待解析字段的路径	否	@message	最大长度：128 最大嵌套键深度：3
分隔符	用于分隔原始逗号分隔值日志事件中的每列的字符	否	,	最大长度：1，除非值为 \t 或 \s
quoteCharacter	用作单列数据的文本限定符的字符	否	"	最大长度：1
列	转换后的日志事件中列要使用的名称列表。	否	[column_1, column_2]	CSV 最大列数：100 最大长度：128 最大嵌套键深度：3

将 delimiter 设置为 \t 将使用制表符分隔每列，\s 将使用单个空格字符分隔每列。

示例：

假设摄取的部分日志事件如下所示：

```
'Akua Mansa':28:'New York: USA'
```

假设我们仅使用 csv 处理器：

```
[
  {
    "csv": {
      "delimiter": ":",
      "quoteCharacter": ""
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "column_1": "Akua Mansa",
  "column_2": "28",
  "column_3": "New York: USA"
}
```

parseKeyValue

使用 parseKeyValue 处理器将指定的字段解析为键值对。您可以使用以下选项自定义处理器以解析字段信息。

字段	描述	必填？	默认值	限制
源	日志事件中待解析字段的路径	否	@message	最大长度：128 最大嵌套键深度：3
目的地	要将提取的键值对放入的目标字段	否		最大长度：128
fieldDelimiter	在原始日志事件的键值对之间使用的字段分隔符字符串	否	&	最大长度：128

字段	描述	必填？	默认值	限制
keyValueDelimiter	在转换后的日志事件中每对的键和值之间使用的分隔符字符串	否	=	最大长度：128
nonMatchValue	当键值对未成功拆分时要在结果的值字段中插入的值。	否		最大长度：128
keyPrefix	如果要为所有转换后的键添加前缀，请在此处指定。	否		最大长度：128
overwriteIfExists	如果目标键已存在，是否覆盖该键	否	false	

示例：

以以下日志事件为例：

```
key1:value1!key2:value2!key3:value3!key4
```

假设我们使用下面的处理器配置：

```
[
  {
    "parseKeyValue": {
      "destination": "new_key",
      "fieldDelimiter": "!",
      "keyValueDelimiter": ":",
      "nonMatchValue": "defaultValue",
      "keyPrefix": "parsed_"
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "new_key": {
    "parsed_key1": "value1",
    "parsed_key2": "value2",
```

```
"parsed_key3": "value3",  
"parsed_key4": "defaultValue"  
}  
}
```

用于 Amazon 公开发布的日志的内置处理器

parseWAF

使用此处理器来解析 Amazon WAF 公开发布的日志，它获取 `httpRequest.headers` 的内容，并根据每个标头名称创建 JSON 键及其对应的值。它还会对 `labels` 执行相同的操作。这些转换可以使查询 Amazon WAF 日志更加容易。有关 Amazon WAF 日志格式的更多信息，请参阅 [Web ACL 流量的日志示例](#)。

此处理器仅接受 `@message` 作为输入。

Important

如果您使用此处理器，则它必须是转换器中的第一个处理器。

示例：

以以下日志事件为例：

```
{  
  "timestamp": 1576280412771,  
  "formatVersion": 1,  
  "webaclId": "arn:aws:wafv2:ap-southeast-2:111122223333:regional/webacl/  
STMTest/1EXAMPLE-2ARN-3ARN-4ARN-123456EXAMPLE",  
  "terminatingRuleId": "STMTest_SQLi_XSS",  
  "terminatingRuleType": "REGULAR",  
  "action": "BLOCK",  
  "terminatingRuleMatchDetails": [  
    {  
      "conditionType": "SQL_INJECTION",  
      "sensitivityLevel": "HIGH",  
      "location": "HEADER",  
      "matchedData": ["10", "AND", "1"]  
    }  
  ],  
  "httpSourceName": "-",
```

```
"httpSourceId": "-",
"ruleGroupList": [],
"rateBasedRuleList": [],
"nonTerminatingMatchingRules": [],
"httpRequest": {
  "clientId": "1.1.1.1",
  "country": "AU",
  "headers": [
    { "name": "Host", "value": "localhost:1989" },
    { "name": "User-Agent", "value": "curl/7.61.1" },
    { "name": "Accept", "value": "*/*" },
    { "name": "x-stm-test", "value": "10 AND 1=1" }
  ],
  "uri": "/myUri",
  "args": "",
  "httpVersion": "HTTP/1.1",
  "httpMethod": "GET",
  "requestId": "rid"
},
"labels": [{ "name": "value" }]
}
```

处理器配置如下所示：

```
[
  {
    "parseWAF": {}
  }
]
```

转换后的日志事件将如下所示。

```
{
  "httpRequest": {
    "headers": {
      "Host": "localhost:1989",
      "User-Agent": "curl/7.61.1",
      "Accept": "*/*",
      "x-stm-test": "10 AND 1=1"
    },
    "clientId": "1.1.1.1",
    "country": "AU",
    "uri": "/myUri",
```

```

    "args": "",
    "httpVersion": "HTTP/1.1",
    "httpMethod": "GET",
    "requestId": "rid"
  },
  "labels": { "name": "value" },
  "timestamp": 1576280412771,
  "formatVersion": 1,
  "webaclId": "arn:aws:wafv2:ap-southeast-2:111122223333:regional/webacl/
STMTest/1EXAMPLE-2ARN-3ARN-4ARN-123456EXAMPLE",
  "terminatingRuleId": "STMTest_SQLi_XSS",
  "terminatingRuleType": "REGULAR",
  "action": "BLOCK",
  "terminatingRuleMatchDetails": [
    {
      "conditionType": "SQL_INJECTION",
      "sensitivityLevel": "HIGH",
      "location": "HEADER",
      "matchedData": ["10", "AND", "1"]
    }
  ],
  "httpSourceName": "-",
  "httpSourceId": "-",
  "ruleGroupList": [],
  "rateBasedRuleList": [],
  "nonTerminatingMatchingRules": []
}

```

parsePostgres

使用此处理器来解析 Amazon RDS for PostgreSQL 公开发布的日志、提取字段并将其转换为 JSON 格式。有关 RDS for PostgreSQL 日志格式的更多信息，请参阅 [RDS for PostgreSQL 数据库日志文件](#)。

此处理器仅接受 @message 作为输入。

Important

如果您使用此处理器，则它必须是转换器中的第一个处理器。

示例：

以以下日志事件为例：

```
2019-03-10 03:54:59 UTC:10.0.0.123(52834):postgres@logtestdb:[20175]:ERROR: column
"wrong_column_name" does not exist at character 8
```

处理器配置如下所示：

```
[
  {
    "parsePostgres": {}
  }
]
```

转换后的日志事件将如下所示。

```
{
  "logTime": "2019-03-10 03:54:59 UTC",
  "srcIp": "10.0.0.123(52834)",
  "userName": "postgres",
  "dbName": "logtestdb",
  "processId": "20175",
  "logLevel": "ERROR"
}
```

parseCloudfront

使用此处理器来解析 Amazon CloudFront 公开发布的日志、提取字段并将其转换为 JSON 格式。对编码的字段值进行解码。整数和双精度值按此方法处理。有关 Amazon CloudFront 日志格式的更多信息，请参阅[配置和使用标准日志（访问日志）](#)。

此处理器仅接受 @message 作为输入。

Important

如果您使用此处理器，则它必须是转换器中的第一个处理器。

示例：

以以下日志事件为例：

```
2019-12-04 21:02:31 LAX1 392 192.0.2.24 GET
d1111111abcdef8.cloudfront.net /index.html 200 - Mozilla/5.0%20(Windows
```

```
%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,
%20like%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36  - - Hit
    SOX4xwn4XV6Q4rgb7XiVG0Hms_BG1TAC4KyHmureZmBNrjGdRLiNIQ==
    d111111abcdef8.cloudfront.net  https  23 0.001  -  TLSv1.2  ECDHE-RSA-AES128-GCM-
    SHA256  Hit  HTTP/2.0  - - 11040 0.001 Hit  text/html 78 - -
```

处理器配置如下所示：

```
[
  {
    "parseCloudfront": {}
  }
]
```

转换后的日志事件将如下所示。

```
{
  "date": "2019-12-04",
  "time": "21:02:31",
  "x-edge-location": "LAX1",
  "sc-bytes": 392,
  "c-ip": "192.0.2.24",
  "cs-method": "GET",
  "cs(Host)": "d111111abcdef8.cloudfront.net",
  "cs-uri-stem": "/index.html",
  "sc-status": 200,
  "cs(Referer)": "-",
  "cs(User-Agent)": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36",
  "cs-uri-query": "-",
  "cs(Cookie)": "-",
  "x-edge-result-type": "Hit",
  "x-edge-request-id": "SOX4xwn4XV6Q4rgb7XiVG0Hms_BG1TAC4KyHmureZmBNrjGdRLiNIQ==",
  "x-host-header": "d111111abcdef8.cloudfront.net",
  "cs-protocol": "https",
  "cs-bytes": 23,
  "time-taken": 0.001,
  "x-forwarded-for": "-",
  "ssl-protocol": "TLSv1.2",
  "ssl-cipher": "ECDHE-RSA-AES128-GCM-SHA256",
  "x-edge-response-result-type": "Hit",
  "cs-protocol-version": "HTTP/2.0",
  "fle-status": "-",
```



```
"fle-encrypted-fields": "-",
"c-port": 11040,
"time-to-first-byte": 0.001,
"x-edge-detailed-result-type": "Hit",
"sc-content-type": "text/html",
"sc-content-len": 78,
"sc-range-start": "-",
"sc-range-end": "-"
}
```

parseRoute53

使用此处理器来解析 Amazon Route 53 Public Data Plane 公开发布的日志、提取字段并将其转换为 JSON 格式。对编码的字段值进行解码。此处理器不支持 Amazon Route 53 Resolver 日志。

此处理器仅接受 @message 作为输入。

Important

如果您使用此处理器，则它必须是转换器中的第一个处理器。

示例：

以以下日志事件为例：

```
1.0 2017-12-13T08:15:50.235Z Z123412341234 example.com AAAA NOERROR TCP IAD12 192.0.2.0
198.51.100.0/24
```

处理器配置如下所示：

```
[
  {
    "parseRoute53": {}
  }
]
```

转换后的日志事件将如下所示。

```
{
  "version": 1.0,
  "queryTimestamp": "2017-12-13T08:15:50.235Z",
```

```
"hostZoneId": "Z123412341234",
"queryName": "example.com",
"queryType": "AAAA",
"responseCode": "NOERROR",
"protocol": "TCP",
"edgeLocation": "IAD12",
"resolverIp": "192.0.2.0",
"ednsClientSubnet": "198.51.100.0/24"
}
```

parseVPC

使用此处理器来解析 Amazon VPC 公开发布的日志、提取字段并将其转换为 JSON 格式。对编码的字段值进行解码。

此处理器仅接受 @message 作为输入。

Important

如果您使用此处理器，则它必须是转换器中的第一个处理器。

示例：

以以下日志事件为例：

```
2 123456789010 eni-abc123de 192.0.2.0 192.0.2.24 20641 22 6 20 4249 1418530010
1418530070 ACCEPT OK
```

处理器配置如下所示：

```
[
  {
    "parseVPC": {}
  }
]
```

转换后的日志事件将如下所示。

```
{
  "version": 2,
  "accountId": "123456789010",
```

```
"interfaceId": "eni-abc123de",
"srcAddr": "192.0.2.0",
"dstAddr": "192.0.2.24",
"srcPort": 20641,
"dstPort": 22,
"protocol": 6,
"packets": 20,
"bytes": 4249,
"start": 1418530010,
"end": 1418530070,
"action": "ACCEPT",
"logStatus": "OK"
}
```

parseToOCSF

parseToOCSF 处理器将日志转换为开放式网络安全架构框架 (OCSF) 事件。OCSF 是一种开放标准，它为安全数据提供了通用架构，从而能够在不同安全工具和平台之间实现更好的互操作性和分析。

此处理器对于安全分析工作流尤其有用，在这些工作流中，您需要将各种 Amazon 服务的日志格式标准化为一致的架构以进行下游分析。

参数

eventSource (必需)

指定生成要转换的日志事件的 Amazon 服务或进程。有效值为：

- CloudTrail - CloudTrail 日志
- Route53Resolver - Route 53 Resolver 日志
- VPCFlow - Amazon VPC 流日志
- EKSAudit - Amazon EKS 审计日志
- AWSWAF - Amazon WAF 日志

ocsfVersion (必需)

指定用于转换后的日志事件的 OCSF 架构版本。目前支持的版本：V1.1

source (可选)

要解析的日志事件中字段的路径。如果省略，则解析整条日志消息。

示例：

以下示例展示了如何使用 `parseToOCSF` 将 VPC 流日志转换为 OCSF 格式：

```
{
  "parseToOCSF": {
    "eventSource": "VPCFlow",
    "ocsfVersion": "V1.1"
  }
}
```

字符串变异处理器

`lowerCaseString`

`lowerCaseString` 处理器会将字符串转换为其小写版本。

字段	描述	必填？	默认值	限制
<code>withKeys</code>	要转换为小写的键列表	是		最多条目数：10

示例：

以以下日志事件为例：

```
{
  "outer_key": {
    "inner_key": "INNER_VALUE"
  }
}
```

转换器配置如下，将 `lowerCaseString` 和 `parseJSON` 结合使用：

```
[
  {
    "parseJSON": {}
  },
  {
    "lowerCaseString": {
      "withKeys": ["outer_key.inner_key"]
    }
  }
]
```

```
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

upperCaseString

upperCaseString 处理器会将字符串转换为其大写版本。

字段	描述	必填？	默认值	限制
withKeys	要转换为大写的键列表	是		最多条目数：10

示例：

以以下日志事件为例：

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

转换器配置如下，将 upperCaseString 和 parseJSON 结合使用：

```
[
  {
    "parseJSON": {}
  },
  {
    "upperCaseString": {
      "withKeys": ["outer_key.inner_key"]
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": "INNER_VALUE"
  }
}
```

splitString

splitString 处理器是一种字符串变异处理器，它使用分隔字符将字段拆分为数组。

字段	描述	必填？	默认值	限制
entries	条目数组。数组中的每个项目都必须包含 source 和 delimiter 字段。	是		最多条目数：10
源	要拆分的字段值的键	是		最大长度：128
分隔符	用于拆分字段值的分隔符字符串	是		最大长度：128

示例 1：

以以下日志事件为例：

```
[
  {
    "parseJSON": {}
  },
  {
    "splitString": {
      "entries": [
        {
          "source": "outer_key.inner_key",
          "delimiter": "_"
        }
      ]
    }
  }
]
```

```
]
```

转换器配置如下，将 `splitString` 和 `parseJSON` 结合使用：

```
[
  {
    "parseJSON": {}
  },
  {
    "splitString": {
      "entries": [
        {
          "source": "outer_key.inner_key",
          "delimiter": "_"
        }
      ]
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": [
      "inner",
      "value"
    ]
  }
}
```

示例 2：

用于拆分字符串的分隔符可以包含多个字符。

以以下日志事件为例：

```
{
  "outer_key": {
    "inner_key": "item1, item2, item3"
  }
}
```

转换器配置如下：

```
[
  {
    "parseJSON": {}
  },
  {
    "splitString": {
      "entries": [
        {
          "source": "outer_key.inner_key",
          "delimiter": ", "
        }
      ]
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": [
      "item1",
      "item2",
      "item3"
    ]
  }
}
```

substituteString

substituteString 处理器是一种字符串变异处理器，它将键的值与正则表达式进行匹配，并将所有匹配项替换为替换字符串。

字段	描述	必填？	默认值	限制
entries	条目数组。数组中的每个项目都必须包含 source、from 和 to 字段。	是		最多条目数：10

字段	描述	必填？	默认值	限制
源	要修改的字段的键	是		最大长度：128 最大嵌套键深度：3
from	要替换的正则表达式字符串。特殊正则表达式字符（例如 [和] ）在使用双引号时必须使用 \ 进行转义，在使用单引号时或通过 Amazon Web Services 管理控制台配置时必须使用 \ 进行转义。有关更多信息，请参阅 Oracle 网站上的 Class Pattern。 您可以将模式封装在 (...) 中以创建带编号的捕获组，并创建可在 to 字段中引用的 (?P<group_name>...) 命名捕获组。	是		最大长度：128
到	对于捕获组的 from 反向引用的每个匹配项，可以使用要替换的字符串。对编号组使用 \$n 形式（如 \$1 ），对命名组使用 \${group_name} （如 \${my_group} ）。	是		最大长度：128 反向引用最大数量：10 重复反向引用最大数量：2

示例 1：

以以下日志事件为例：

```
{
  "outer_key": {
    "inner_key1": "[]",
    "inner_key2": "123-345-567",
    "inner_key3": "A cat takes a catnap."
  }
}
```

转换器配置如下，将 `substituteString` 和 `parseJSON` 结合使用：

```
[
  {
    "parseJSON": {}
  },
  {
    "substituteString": {
      "entries": [
        {
          "source": "outer_key.inner_key1",
          "from": "\\[\\]",
          "to": "value1"
        },
        {
          "source": "outer_key.inner_key2",
          "from": "[0-9]{3}-[0-9]{3}-[0-9]{3}",
          "to": "xxx-xxx-xxx"
        },
        {
          "source": "outer_key.inner_key3",
          "from": "cat",
          "to": "dog"
        }
      ]
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key1": "value1",
    "inner_key2": "xxx-xxx-xxx",
    "inner_key3": "A dog takes a dognap."
  }
}
```

示例 2：

以以下日志事件为例：

```
{
  "outer_key": {
    "inner_key1": "Tom, Dick, and Harry",
    "inner_key2": "arn:aws:sts::123456789012:assumed-role/MyImportantRole/MySession"
  }
}
```

转换器配置如下，将 `substituteString` 和 `parseJSON` 结合使用：

```
[
  {
    "parseJSON": {}
  },
  {
    "substituteString": {
      "entries": [
        {
          "source": "outer_key.inner_key1",
          "from": "(\\w+), (\\w+), and (\\w+)",
          "to": "$1 and $3"
        },
        {
          "source": "outer_key.inner_key2",
          "from": "^arn:aws:sts::(?P<account_id>\\d{12}):assumed-role/(?P<role_name>[\\w+=,.-]+)/(?P<role_session_name>[\\w+=,.-]+)$",
          "to": "${account_id}:${role_name}:${role_session_name}"
        }
      ]
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key1": "Tom and Harry",
    "inner_key2": "123456789012:MyImportantRole:MySession"
  }
}
```

trimString

trimString 处理器会移除键开头和结尾的空格。

字段	描述	必填？	默认值	限制
withKeys	要剪裁的键列表	是		最多条目数：10

示例：

以以下日志事件为例：

```
{
  "outer_key": {
    "inner_key": "  inner_value  "
  }
}
```

转换器配置如下，将 trimString 和 parseJSON 结合使用：

```
[
  {
    "parseJSON": {}
  },
  {
    "trimString": {
      "withKeys": ["outer_key.inner_key"]
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

JSON 变异处理器

addKeys

使用 addKeys 处理器向日志事件添加新的键值对。

字段	描述	必填？	默认值	限制
entries	条目数组。数组中的每个项目都可以包含 key、value 和 overwriteIfExists 字段。	是		最多条目数：5
键	要添加的新条目的键	是		最大长度：128 最大嵌套键深度：3
值	要添加的新条目的值	是		最大长度：256
overwriteIfExists	如果您将其设置为 true，而事件中已存在 key，则现有值会被覆盖。默认值为 false。	否	false	无限制

示例：

以以下日志事件为例：

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

转换器配置如下，将 addKeys 和 parseJSON 结合使用：

```
[
  {
    "parseJSON": {}
  },
  {
```

```
      "addKeys": {
        "entries": [
          {
            "source": "outer_key.new_key",
            "value": "new_value"
          }
        ]
      }
    }
  ]
}
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": "inner_value",
    "new_key": "new_value"
  }
}
```

deleteKeys

使用 deleteKeys 处理器从日志事件中删除字段。这些字段可以包含键值对。

字段	描述	必填？	默认值	限制
withKeys	要删除的键列表。	是	无限制	最多条目数：5

示例：

以以下日志事件为例：

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

转换器配置如下，将 deleteKeys 和 parseJSON 结合使用：

```
[
  {
    "parseJSON": {}
  },
  {
    "deleteKeys": {
      "withKeys": ["outer_key.inner_key"]
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {}
}
```

moveKeys

使用 moveKeys 处理器将键从一个字段移动到另一个字段。

字段	描述	必填？	默认值	限制
entries	条目数组。数组中的每个项目都可以包含 source、target 和 overwriteIfExists 字段。	是		最多条目数：5
源	要移动的键	是		最大长度：128 最大嵌套键深度：3
target	要移动至的键	是		最大长度：128 最大嵌套键深度：3
overwriteIfExists	如果您将其设置为 true，而事件中已存在 key，则现有值会被覆盖。默认值为 false。	否	false	无限制

示例：

以以下日志事件为例：

```
{
  "outer_key1": {
    "inner_key1": "inner_value1"
  },
  "outer_key2": {
    "inner_key2": "inner_value2"
  }
}
```

转换器配置如下，将 moveKeys 和 parseJSON 结合使用：

```
[
  {
    "parseJSON": {}
  },
  {
    "moveKeys": {
      "entries": [
        {
          "source": "outer_key1.inner_key1",
          "target": "outer_key2"
        }
      ]
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key1": {},
  "outer_key2": {
    "inner_key2": "inner_value2",
    "inner_key1": "inner_value1"
  }
}
```

renameKeys

使用 renameKeys 处理器重命名日志事件中的键。

字段	描述	必填？	默认值	限制
entries	条目数组。数组中的每个项目都可以包含 key、target 和 overwriteIfExists 字段。	是	无限制	最多条目数：5
键	要重命名的键	是	无限制	最大长度：128
target	新的键名称	是	无限制	最大长度：128 最大嵌套键深度：3
overwriteIfExists	如果您将其设置为 true，而事件中已存在 key，则现有值会被覆盖。默认值为 false。	否	false	无限制

示例：

以以下日志事件为例：

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

转换器配置如下，将 renameKeys 和 parseJSON 结合使用：

```
[
  {
    "parseJSON": {}
  },
  {
    "renameKeys": {
      "entries": [
        {
          "key": "outer_key",
          "target": "new_key"
        }
      ]
    }
  }
]
```

```
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "new_key": {
    "inner_key": "inner_value"
  }
}
```

copyValue

使用 copyValue 处理器复制日志事件中的值。您还可以通过将以下元数据键的值复制到日志事件中，来使用此处理器向日志事件添加元数据：@logGroupName、@logGroupStream、@accountId、@regionName。以下示例对此进行了说明。

字段	描述	必填？	默认值	限制
entries	条目数组。数组中的每个项目都可以包含 source、target 和 overwriteIfExists 字段。	是		最多条目数：5
源	要复制的键	是		最大长度：128 最大嵌套键深度：3
target	将值复制到的键	是	无限制	最大长度：128 最大嵌套键深度：3
overwriteIfExists	如果您将其设置为 true，而事件中已存在 key，则现有值会被覆盖。默认值为 false。	否	false	无限制

示例：

以以下日志事件为例：

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

转换器配置如下，将 copyValue 和 parseJSON 结合使用：

```
[
  {
    "parseJSON": {}
  },
  {
    "copyValue": {
      "entries": [
        {
          "source": "outer_key.new_key",
          "target": "new_key"
        },
        {
          "source": "@logGroupName",
          "target": "log_group_name"
        },
        {
          "source": "@logGroupStream",
          "target": "log_group_stream"
        },
        {
          "source": "@accountId",
          "target": "account_id"
        },
        {
          "source": "@regionName",
          "target": "region_name"
        }
      ]
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": "inner_value"
  },
  "new_key": "inner_value",
  "log_group_name": "myLogGroupName",
  "log_group_stream": "myLogStreamName",
  "account_id": "012345678912",
  "region_name": "us-east-1"
}
```

listToMap

listToMap 处理器获取包含键字段的对象列表，并将其转换为目标键的映射。

字段	描述	必填？	默认值	限制
源	ProcessingEvent 中的键，其中包含要转换为映射的对象列表	是		最大长度：128 最大嵌套键深度：3
键	要在生成的映射中提取为键的字段的键	是		最大长度：128
valueKey	如果指定此项，则将从 source 对象中提取您在此参数中指定的值，并将其放入生成的映射的值中。否则，源列表中的原始对象将放入生成的映射的值中。	否		最大长度：128
target	将保存生成的映射的字段的键	否	根节点	最大长度：128 最大嵌套键深度：3
flatten	一个布尔值，指示列表是扁平化为单个项目，还是生成的映射中的值将是列表。 默认情况下，匹配键的值将以数组表示。请将 flatten 设置	否	false	

字段	描述	必填？	默认值	限制
	为 true，以根据 flattened Element 的值将数组转换为单个值。			
flattened Element	如果将 flatten 设置为 true，则使用 flattenedElement 指定要保留哪个元素（first 或 last）。	在 flatten 设置为 true 时是必需的		值只能是 first 或 last

示例：

以以下日志事件为例：

```
{
  "outer_key": [
    {
      "inner_key": "a",
      "inner_value": "val-a"
    },
    {
      "inner_key": "b",
      "inner_value": "val-b1"
    },
    {
      "inner_key": "b",
      "inner_value": "val-b2"
    },
    {
      "inner_key": "c",
      "inner_value": "val-c"
    }
  ]
}
```

使用案例 1 的转换器：flatten 是 false

```
[
  {
```

```
    "parseJSON": {}
  },
  {
    "listToMap": {
      "source": "outer_key"
      "key": "inner_key",
      "valueKey": "inner_value",
      "flatten": false
    }
  }
}
```

转换后的日志事件将如下所示。

```
{
  "outer_key": [
    {
      "inner_key": "a",
      "inner_value": "val-a"
    },
    {
      "inner_key": "b",
      "inner_value": "val-b1"
    },
    {
      "inner_key": "b",
      "inner_value": "val-b2"
    },
    {
      "inner_key": "c",
      "inner_value": "val-c"
    }
  ],
  "a": [
    "val-a"
  ],
  "b": [
    "val-b1",
    "val-b2"
  ],
  "c": [
    "val-c"
  ]
}
```

```
}
```

使用案例 2 的转换器：flatten 是 true，flattenedElement 是 first

```
[
  {
    "parseJSON": {}
  },
  {
    "listToMap": {
      "source": "outer_key"
      "key": "inner_key",
      "valueKey": "inner_value",
      "flatten": true,
      "flattenedElement": "first"
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": [
    {
      "inner_key": "a",
      "inner_value": "val-a"
    },
    {
      "inner_key": "b",
      "inner_value": "val-b1"
    },
    {
      "inner_key": "b",
      "inner_value": "val-b2"
    },
    {
      "inner_key": "c",
      "inner_value": "val-c"
    }
  ],
  "a": "val-a",
  "b": "val-b1",
  "c": "val-c"
}
```

```
}
```

使用案例 3 的转换器：flatten 是 true，flattenedElement 是 last

```
[
  {
    "parseJSON": {}
  },
  {
    "listToMap": {
      "source": "outer_key"
      "key": "inner_key",
      "valueKey": "inner_value",
      "flatten": true,
      "flattenedElement": "last"
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": [
    {
      "inner_key": "a",
      "inner_value": "val-a"
    },
    {
      "inner_key": "b",
      "inner_value": "val-b1"
    },
    {
      "inner_key": "b",
      "inner_value": "val-b2"
    },
    {
      "inner_key": "c",
      "inner_value": "val-c"
    }
  ],
  "a": "val-a",
  "b": "val-b2",
  "c": "val-c"
}
```



```
}

```

数据类型转换器处理器

typeConverter

使用 typeConverter 处理器可以将与指定的键关联的值类型转换为指定的类型。它是一个转换处理器，可以更改指定字段的类型。值可以转换为以下数据类型之一：integer、double、string 和 boolean。

字段	描述	必填？	默认值	限制
entries	条目数组。数组中的每个项目都必须包含 key 和 type 字段。	是		最多条目数：10
键	要转换为其他类型的键和值	是		最大长度：128 最大嵌套键深度：3
type	要转换为的类型。有效值包括 integer、double、string 和 boolean。	是		

示例：

以以下日志事件为例：

```
{
  "name": "value",
  "status": "200"
}

```

转换器配置如下，将 typeConverter 和 parseJSON 结合使用：

```
[
  {
    "parseJSON": {}
  },
  {

```

```
    "typeConverter": {
      "entries": [
        {
          "key": "status",
          "type": "integer"
        }
      ]
    }
  }
}
```

转换后的日志事件将如下所示。

```
{
  "name": "value",
  "status": 200
}
```

datetimeConverter

使用 datetimeConverter 处理器可以将日期时间字符串转换为您指定的格式。

字段	描述	必填？	默认值	限制
源	对其应用日期转换的键。	是		最多条目数：10
matchPatterns	与 source 字段匹配的模式列表	是		最多条目数：5
target	用于存储结果的 JSON 字段。	是		最大长度：128 最大嵌套键深度：3
targetFormat	用于目标字段中的转换后数据的日期时间格式。	否	yyyy-MM-dd'T'HH:mm:ss.SSS'Z'	最大长度：64
sourceTimezone	源字段的时区。	否	UTC	最小长度：1

字段	描述	必填？	默认值	限制
	有关可能值的列表，请参阅 Java Supported Zone Ids and Offsets 。			
目标时区	目标字段的时区。 有关可能值的列表，请参阅 Java Supported Zone Ids and Offsets 。	否	UTC	最小长度：1
locale	源字段的区域设置。 有关可能值的列表，请参阅 Locale getAvailableLocales() Method in Java with Examples 。	是		最小长度：1

示例：

以以下日志事件为例：

```
{"german_datetime": "Samstag 05. Dezember 1998 11:00:00"}
```

转换器配置如下，将 dateTimeConverter 和 parseJSON 结合使用：

```
[
  {
    "parseJSON": {}
  },
  {
    "dateTimeConverter": {
      "source": "german_datetime",
      "target": "target_1",
      "locale": "de",
      "matchPatterns": ["EEEE dd. MMMM yyyy HH:mm:ss"],
      "sourceTimezone": "Europe/Berlin",
      "targetTimezone": "America/New_York",
      "targetFormat": "yyyy-MM-dd'T'HH:mm:ss z"
    }
  }
]
```

转换后的日志事件将如下所示。

```
{
  "german_datetime": "Samstag 05. Dezember 1998 11:00:00",
  "target_1": "1998-12-05T17:00:00 MEZ"
}
```

转换指标和错误

CloudWatch Logs 会将转换指标发布到 CloudWatch。这些指标包括 TransformedLogEvents、TransformedBytes 和 TransformationErrors。有关更多信息，请参阅 [日志转换器指标和维度](#)。

每当 CloudWatch Logs 尝试转换日志事件但失败时，它都会向该日志事件添加一个 @transformationError 系统字段。运行 CloudWatch Logs Insights 查询时，您将在所有包含转换失败的日志事件中看到此字段。您可以使用一个查询来查询此字段，例如 `filter ispresent(@transformationError)` 可查找所有失败的转换事件。

使用 Amazon OpenSearch Service 进行分析

CloudWatch Logs 与 Amazon OpenSearch Service 集成，使您能够创建自动精选控制面板，其显示 OpenSearch Service 从 Amazon 公开发布的日志中获取的关键指标。以下控制面板可用：

- Amazon VPC 流日志控制面板捕获 Amazon VPC 的网络流量数据。它帮助您分析网络流量、检测异常模式和监控资源使用情况。显示的关键指标包括：
 - 总流量以及这些流量的接受和拒绝情况
 - 一段时间内的流量模式
 - 桑基图，展示源 IP 和目标 IP 之间的数据流（主要贡献者）
 - 按传输字节数和数据包数排名的前几大 IP

Note

目前仅支持 VPC 版本 2 字段格式。

- Amazon WAF 日志控制面板可让您深入了解 Amazon WAF 监控的 Web 流量。此控制面板可帮助您识别来自特定区域或 IP 的流量模式、被阻止的请求和潜在威胁。显示的关键指标包括：
 - 总请求数，包括“允许”和“阻止”计数。
 - 一段时间内的请求历史记录，显示允许和阻止的请求。
 - 按 Web ACL 名称、终止规则阻止的请求和源 IP 划分的请求明细。
 - 请求源的地理分布。
 - 按请求计数排名的前几大客户端 IP 和终止规则。
- CloudTrail 日志控制面板概述了您的 Amazon 环境中使用 CloudTrail 日志的 API 活动。它可用于监控 API 活动、审计操作以及识别潜在的安全或合规性问题。显示的关键指标包括：
 - 一段时间内的事件总数和事件历史记录
 - 按账户 ID、类别和区域划分的事件明细。
 - 生成事件所涉及的热门 API、服务和源 IP。
 - 生成事件的热门用户的表，详细说明用户账户信息和事件计数。
- Amazon Network Firewall 控制面板可增强网络流量的可见性，为安全监控和分析提供宝贵的见解。此控制面板提供各种网络指标和模式的全面视图，以便快速识别潜在安全问题并优化网络配置。显示的关键指标包括：
 - 主要贡献者和协议

- 对 PrivateLink 端点的见解
- 允许和阻止的 TLS 服务器名称指示流量

这些精选控制面板中显示的指标均来自 Amazon OpenSearch Service 分析。

在查看这些控制面板之前，您必须创建一个 IAM 角色，并将 CloudWatch Logs 与 Amazon OpenSearch Service 进行一次性集成。这种一次性集成会配置创建和呈现控制面板所需的 Amazon OpenSearch Service 资源。您需要为使用的 OpenSearch 服务付费。有关更多信息，请参阅 [Amazon CloudWatch 定价](#)。

您只能为标准日志类中的日志组创建这些精选控制面板。

Important

不要对任何需要创建公开发布的日志控制面板的日志组使用 [日志转换器](#)。转换日志事件将导致控制面板的数据为空。

主题

- [步骤 1：创建与 OpenSearch Service 的集成](#)
- [步骤 2：创建公开发布的日志控制面板](#)
- [查看、编辑或删除公开发布的日志控制面板](#)
- [针对用户的 IAM 策略](#)
- [集成所需的权限](#)

步骤 1：创建与 OpenSearch Service 的集成

第一步是创建与 OpenSearch Service 的集成，只需执行一次。创建集成将在您的账户中创建以下资源。

- 不具备高可用性的 [OpenSearch Service 时间序列集合](#)。

集合是一组 OpenSearch Service 索引，它们协同工作以支持工作负载。

- 集合的两个安全策略。一个策略定义加密类型，可以使用客户管理型 Amazon KMS 密钥或服务拥有的密钥。另一个策略定义网络访问权限，允许 OpenSearch Service 应用程序访问集合。有关更多信息，请参阅 [Amazon OpenSearch Service 中的静态数据加密](#)。

- [OpenSearch Service 数据访问策略](#)，用于定义哪些用户可以访问集合中的数据。
- [OpenSearch Service 直接查询数据来源](#)，将 CloudWatch Logs 定义为源。
- 名为 aws-analytics 的 [OpenSearch Service 应用程序](#)。该应用程序将配置为允许创建工作区。如果名为 aws-analytics 的应用程序已存在，则会对其进行更新，以添加此集合作为数据来源。
- [OpenSearch Service 工作区](#)，将托管控制面板，并允许所有被授予访问权限的人从工作区读取。

主题

- [所需的权限](#)
- [创建集成](#)

所需的权限

要创建集成，您必须登录到拥有 CloudWatchOpenSearchDashboardsFullAccess 托管 IAM 策略或等效权限的账户（如此处所示）。您还必须拥有这些权限才能删除集成、创建、编辑和删除控制面板以及手动刷新控制面板。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "CloudWatchOpenSearchDashboardsIntegration",
    "Effect": "Allow",
    "Action": [
      "logs:ListIntegrations",
      "logs:GetIntegration",
      "logs>DeleteIntegration",
      "logs:PutIntegration",
      "logs:DescribeLogGroups",
      "opensearch:ApplicationAccessAll",
      "iam:ListRoles",
      "iam:ListUsers"
    ],
    "Resource": "*"
  },
  {
    "Sid": "CloudWatchLogsOpensearchReadAPIs",
    "Effect": "Allow",
```

```

        "Action": [
            "aoss:BatchGetCollection",
            "aoss:BatchGetLifecyclePolicy",
            "es:ListApplications"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:CalledViaFirst": "logs.amazonaws.com"
            }
        }
    },
    {
        "Sid": "CloudWatchLogsOpensearchCreateServiceLinkedAccess",
        "Effect": "Allow",
        "Action": [
            "iam:CreateServiceLinkedRole"
        ],
        "Resource": "arn:aws:iam::*:role/aws-service-role/
opensearchservice.amazonaws.com/AWSServiceRoleForAmazonOpenSearchService",
        "Condition": {
            "StringEquals": {
                "iam:AWSServiceName": "opensearchservice.amazonaws.com",
                "aws:CalledViaFirst": "logs.amazonaws.com"
            }
        }
    },
    {
        "Sid": "CloudWatchLogsObservabilityCreateServiceLinkedAccess",
        "Effect": "Allow",
        "Action": [
            "iam:CreateServiceLinkedRole"
        ],
        "Resource": "arn:aws:iam::*:role/aws-service-role/
observability.aoss.amazonaws.com/AWSServiceRoleForAmazonOpenSearchServerless",
        "Condition": {
            "StringEquals": {
                "iam:AWSServiceName": "observability.aoss.amazonaws.com",
                "aws:CalledViaFirst": "logs.amazonaws.com"
            }
        }
    },
    {
        "Sid": "CloudWatchLogsCollectionRequestAccess",

```



```

    "Effect": "Allow",
    "Action": [
        "aoss:CreateCollection"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:CalledViaFirst": "logs.amazonaws.com",
            "aws:RequestTag/CloudWatchOpenSearchIntegration": [
                "Dashboards"
            ]
        },
        "ForAllValues:StringEquals": {
            "aws:TagKeys": "CloudWatchOpenSearchIntegration"
        }
    }
},
{
    "Sid": "CloudWatchLogsApplicationRequestAccess",
    "Effect": "Allow",
    "Action": [
        "es:CreateApplication"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:CalledViaFirst": "logs.amazonaws.com",
            "aws:RequestTag/OpenSearchIntegration": [
                "Dashboards"
            ]
        },
        "ForAllValues:StringEquals": {
            "aws:TagKeys": "OpenSearchIntegration"
        }
    }
},
{
    "Sid": "CloudWatchLogsCollectionResourceAccess",
    "Effect": "Allow",
    "Action": [
        "aoss:DeleteCollection"
    ],
    "Resource": "*",
    "Condition": {

```

```

        "StringEquals": {
            "aws:CalledViaFirst": "logs.amazonaws.com",
            "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                "Dashboards"
            ]
        }
    },
    {
        "Sid": "CloudWatchLogsApplicationResourceAccess",
        "Effect": "Allow",
        "Action": [
            "es:UpdateApplication",
            "es:GetApplication"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:CalledViaFirst": "logs.amazonaws.com",
                "aws:ResourceTag/OpenSearchIntegration": [
                    "Dashboards"
                ]
            }
        }
    },
    {
        "Sid": "CloudWatchLogsCollectionPolicyAccess",
        "Effect": "Allow",
        "Action": [
            "aoss:CreateSecurityPolicy",
            "aoss:CreateAccessPolicy",
            "aoss:DeleteAccessPolicy",
            "aoss:DeleteSecurityPolicy",
            "aoss:GetAccessPolicy",
            "aoss:GetSecurityPolicy"
        ],
        "Resource": "*",
        "Condition": {
            "StringLike": {
                "aoss:collection": "cloudwatch-logs-*",
                "aws:CalledViaFirst": "logs.amazonaws.com"
            }
        }
    }
},

```

```

{
  "Sid": "CloudWatchLogsAPIAccessAll",
  "Effect": "Allow",
  "Action": [
    "aoss:APIAccessAll"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "aoss:collection": "cloudwatch-logs-*"
    }
  }
},
{
  "Sid": "CloudWatchLogsIndexPolicyAccess",
  "Effect": "Allow",
  "Action": [
    "aoss:CreateAccessPolicy",
    "aoss:DeleteAccessPolicy",
    "aoss:GetAccessPolicy",
    "aoss:CreateLifecyclePolicy",
    "aoss:DeleteLifecyclePolicy"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "aoss:index": "cloudwatch-logs-*",
      "aws:CalledViaFirst": "logs.amazonaws.com"
    }
  }
},
{
  "Sid": "CloudWatchLogsDQSRequestQueryAccess",
  "Effect": "Allow",
  "Action": [
    "es:AddDirectQueryDataSource"
  ],
  "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*",
  "Condition": {
    "StringEquals": {
      "aws:CalledViaFirst": "logs.amazonaws.com",
      "aws:RequestTag/CloudWatchOpenSearchIntegration": [
        "Dashboards"
      ]
    }
  }
}

```

```

        },
        "ForAllValues:StringEquals": {
            "aws:TagKeys": "CloudWatchOpenSearchIntegration"
        }
    },
    {
        "Sid": "CloudWatchLogsStartDirectQueryAccess",
        "Effect": "Allow",
        "Action": [
            "opensearch:StartDirectQuery",
            "opensearch:GetDirectQuery"
        ],
        "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*"
    },
    {
        "Sid": "CloudWatchLogsDQSResourceQueryAccess",
        "Effect": "Allow",
        "Action": [
            "es:GetDirectQueryDataSource",
            "es>DeleteDirectQueryDataSource"
        ],
        "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*",
        "Condition": {
            "StringEquals": {
                "aws:CalledViaFirst": "logs.amazonaws.com",
                "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                    "Dashboards"
                ]
            }
        }
    },
    {
        "Sid": "CloudWatchLogsPassRoleAccess",
        "Effect": "Allow",
        "Action": [
            "iam:PassRole"
        ],
        "Resource": "*",
        "Condition": {
            "StringLike": {
                "iam:PassedToService":
                "directquery.opensearchservice.amazonaws.com",
                "aws:CalledViaFirst": "logs.amazonaws.com"
            }
        }
    }
}

```

```

    }
  },
  {
    "Sid": "CloudWatchLogsAossTagsAccess",
    "Effect": "Allow",
    "Action": [
      "aoss:TagResource"
    ],
    "Resource": "arn:aws:aoss:*:*:collection/*",
    "Condition": {
      "StringEquals": {
        "aws:CalledViaFirst": "logs.amazonaws.com",
        "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
          "Dashboards"
        ]
      },
      "ForAllValues:StringEquals": {
        "aws:TagKeys": "CloudWatchOpenSearchIntegration"
      }
    }
  },
  {
    "Sid": "CloudWatchLogsEsApplicationTagsAccess",
    "Effect": "Allow",
    "Action": [
      "es:AddTags"
    ],
    "Resource": "arn:aws:opensearch:*:*:application/*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/OpenSearchIntegration": [
          "Dashboards"
        ],
        "aws:CalledViaFirst": "logs.amazonaws.com"
      },
      "ForAllValues:StringEquals": {
        "aws:TagKeys": "OpenSearchIntegration"
      }
    }
  },
  {
    "Sid": "CloudWatchLogsEsDataSourceTagsAccess",
    "Effect": "Allow",

```

```
    "Action": [
      "es:AddTags"
    ],
    "Resource": "arn:aws:opensearch:*:*:datasource/*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
          "Dashboards"
        ],
        "aws:CalledViaFirst": "logs.amazonaws.com"
      },
      "ForAllValues:StringEquals": {
        "aws:TagKeys": "CloudWatchOpenSearchIntegration"
      }
    }
  }
]
```

创建集成

请按照以下步骤创建集成。

将 CloudWatch Logs 与 Amazon OpenSearch Service 集成

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在左侧导航窗格中，选择 Logs Insights，然后选择使用 OpenSearch 进行分析选项卡。
3. 选择创建集成。
4. 在集成名称中，输入集成的名称。
5. （可选）要加密写入 OpenSearch Service Serverless 的数据，请在 KMS 密钥 ARN 中输入要使用的 Amazon KMS 密钥的 ARN。有关更多信息，请参阅《Amazon OpenSearch Service 开发人员指南》中的[静态加密](#)。
6. 对于数据留存，请输入您希望保留 OpenSearch Service 数据索引的时间长度。这同时也定义了您可以在控制面板中查看数据的最长时间段。选择较长的数据留存期将产生额外的搜索和索引费用。有关更多信息，请参阅 [OpenSearch Service 无服务器定价](#)。

最长保留期为 30 天。

数据留存期还将用于创建 OpenSearch Service 集合生命周期策略。

7. 对于用于写入 OpenSearch 集合的 IAM 角色，请创建一个新的 IAM 角色或选择一个现有 IAM 角色来用于写入 OpenSearch Service 集合。

创建新角色是最简单的方法，将创建拥有必要权限的角色。

 Note

如果您创建了一个角色，该角色将拥有读取账户中所有日志组的权限。

如果您想要选择一个现有角色，则该角色应具备[集成所需的权限](#)中列出的权限。或者，您可以选择使用现有角色，然后在验证所选角色的访问权限部分中选择创建角色。这样，您可以将[集成所需的权限](#)中列出的权限用作模板并进行修改。例如，如果您想对日志组进行更精细的控制。

8. 对于可以查看控制面板的 IAM 角色和用户，您可以选择如何向 IAM 角色和 IAM 用户授予访问公开发布的日志控制面板的权限：
 - 要将控制面板访问权限限制为仅限部分用户，请选择选择可以查看控制面板的 IAM 角色和用户，然后在文本框中搜索并选择要授予访问权限的 IAM 角色和 IAM 用户。
 - 要授予所有用户访问控制面板的权限，请选择允许此账户中的所有角色和用户查看控制面板。

 Important

选择角色或用户或者选择所有用户，只会将他们添加到访问存储控制面板数据的 OpenSearch Service 集合所需的[数据访问策略](#)中。要让他们能够查看公开发布的日志控制面板，您还必须授予这些角色和用户 [CloudWatchOpenSearchDashboardAccess](#) 托管 IAM 策略。

9. 选择创建集成

创建集成需要几分钟时间。

步骤 2：创建公开发布的日志控制面板

创建集成后，您可以创建控制面板。控制面板可用于 Amazon VPC 流日志、CloudTrail 日志和 Amazon WAF 日志。

使用包含 OpenSearch Service 派生的指标的公开发布的日志控制面板

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。

2. 在左侧导航窗格中，选择 Logs Insights，然后选择使用 OpenSearch 进行分析选项卡。
3. 请选择创建控制面板。
4. 选择要为其创建控制面板的日志类型：Amazon WAF、Amazon VPC 流日志、CloudTrail 或 Amazon Network Firewall。
5. 输入控制面板名称，并选择性地输入描述。
6. 对于数据同步频率，输入您希望 OpenSearch Service 查询 CloudWatch 的频率，以便将 OpenSearch Service 中创建的指标和索引与新数据同步并更新。OpenSearch Service 会根据您的日志创建指标和索引，以便呈现控制面板。

选择较短的时间可以使数据保持最新状态，但会产生更高的成本。

7. 为此控制面板选择要从中收集数据的日志组。请务必选择与您正在创建的控制面板类型匹配的日志组。

在进行这些选择时，您可以使用浏览日志组按钮和查看选定日志组中的日志示例选项，以确保您获得所需的日志组。

8. 请选择创建控制面板。

控制面板最初显示时没有任何数据。几分钟后，数据才会出现在控制面板中。首次显示数据时，将显示最近 15 分钟的日志条目。

查看、编辑或删除公开发布的日志控制面板

在 CloudWatch Logs 或 OpenSearch Service 中查看公开发布的日志控制面板

要能够查看控制面板，您必须登录到具有 CloudWatchOpenSearchDashboardAccess IAM 策略的 IAM 主体。

查看公开发布的日志控制面板

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在左侧导航窗格中，选择 Logs Insights，然后选择使用 OpenSearch 进行分析选项卡。
3. 在 OpenSearch 控制面板框中选择控制面板。
4. （可选）在右上角选择在 OpenSearch 中查看。

OpenSearch Service 控制台将打开，您将在那里看到相同的控制面板。在 OpenSearch Service 控制台中，您可以对控制面板及其小组件进行更改，当您在 CloudWatch Logs 中查看控制面板时，这些更改也会显示。

向其他 IAM 角色或 IAM 用户授予控制面板查看权限

要在创建集成后向其他 IAM 主体授予访问权限，请执行下面的步骤。

向其他 IAM 角色或用户授予公开发布的日志控制面板访问权限

1. 编辑集合的数据访问策略以添加这些角色或用户。有关更多信息，请参阅《OpenSearch Service 开发人员指南》中的 [Amazon OpenSearch Service 无服务器的数据访问控制](#)。
2. 向这些用户授予 CloudWatchOpenSearchDashboardAccess 有关该策略内容的更多信息，请参阅 [CloudWatchOpenSearchDashboardAccess](#)。

编辑控制面板配置

您可以编辑现有公开发布的日志控制面板的名称、描述和同步频率。

编辑公开发布的日志控制面板

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在左侧导航窗格中，选择 Logs Insights，然后选择使用 OpenSearch 进行分析选项卡。
3. 在 OpenSearch 控制面板框中选择控制面板。
4. 选择操作、更改控制面板详细信息。
5. 进行更改，然后选择确认更改。

删除公开发布的日志控制面板

您可以删除公开发布的日志控制面板。如果这样做，则在 OpenSearch Service 集合中创建的控制面板、指标和索引都会被删除。

 Note

删除公开发布的日志控制面板后，请至少等待六个小时，然后再尝试重新创建相同的控制面板。如果您不等待，重新创建的控制面板将无法正常工作。

删除公开发布的日志控制面板

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在左侧导航窗格中，选择 Logs Insights，然后选择使用 OpenSearch 进行分析选项卡。
3. 在 OpenSearch 控制面板框中选择控制面板。
4. 依次选择操作、删除。
5. 输入 **delete** 确认您的决定，然后选择删除。

删除所有与 OpenSearch Service 的公开发布的日志控制面板集成

您可以删除整个 OpenSearch 集成。如果这样做，则所有公开发布的日志控制面板及其中显示的数据都将被删除。

 Important

为避免持续产生费用，我们强烈建议您在删除集成之前手动删除以下资源。删除集成不会自动删除这些资源，并且删除集成后，您将无法访问这些资源来将其删除。要查找要删除的资源名称，请参阅以下过程。

- [– 数据源](#)。
- [集合](#)
- [数据访问策略](#)
- [加密策略](#)
- [网络策略](#)
- [生命周期策略](#)

删除整个与 OpenSearch Service 的公开发布的日志控制面板集成

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。

2. 在左侧导航窗格中，选择 设置。
3. 选择日志选项卡。
4. 在 OpenSearch 集成部分，选择删除集成。

下一个屏幕会显示您应该在删除集成之前删除的 OpenSearch Service 资源的名称。

5. 输入 **delete** 确认您的决定，然后选择删除集成。

针对用户的 IAM 策略

CloudWatch Logs 创建了两个 IAM 策略：CloudWatchOpenSearchDashboardsFullAccess 和 CloudWatchOpenSearchDashboardAccess。下表列出了这些策略中的每个策略启用的操作。

操作	IAM 策略	需要额外权限
创建集成	CloudWatchOpenSearchDashboardsFullAccess	
删除集成	CloudWatchOpenSearchDashboardsFullAccess	
创建控制面板	CloudWatchOpenSearchDashboardsFullAccess	
编辑控制面板	CloudWatchOpenSearchDashboardsFullAccess	
删除控制面板	CloudWatchOpenSearchDashboardsFullAccess	
使用立即同步刷新控制面板	CloudWatchOpenSearchDashboardsFullAccess	
在设置中查看集成	CloudWatchOpenSearchDashboardAccess 或 CloudWatchOpenSearchDashboardsFullAccess	

操作	IAM 策略	需要额外权限
查看控制面板	CloudWatchOpenSearchDashboardAccess 或 CloudWatchOpenSearchDashboardsFullAccess	创建集成时指定角色或用户，或者编辑集合的数据访问策略以添加这些角色或用户。有关更多信息，请参阅《OpenSearch Service 开发人员指南》中的 Amazon OpenSearch Service 无服务器的数据访问控制 。
在 OpenSearch Service 控制台中查看控制面板	CloudWatchOpenSearchDashboardAccess 或 CloudWatchOpenSearchDashboardsFullAccess	创建集成时指定角色或用户，或者编辑集合的数据访问策略以添加这些角色或用户。有关更多信息，请参阅《OpenSearch Service 开发人员指南》中的 Amazon OpenSearch Service 无服务器的数据访问控制 。

集成所需的权限

如果您创建了一个 IAM 角色供集成使用，则该角色必须包含以下权限和信任策略，而不是允许 CloudWatch Logs 创建角色。有关如何创建 IAM 角色的更多信息，请参阅[创建向 Amazon 服务委派权限的角色](#)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchLogsAccess",
      "Effect": "Allow",
      "Action": [
        "logs:StartQuery",
        "logs:GetLogGroupFields",
```

```

        "logs:GetQueryResults"
    ],
    "Resource": [
        "*"
    ]
},
{
    "Sid": "CloudWatchLogsDescribeLogGroupsAccess",
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogGroups"
    ],
    "Resource": "*"
},
{
    "Sid": "AmazonOpenSearchCollectionAccess",
    "Effect": "Allow",
    "Action": [
        "aoss:APIAccessAll"
    ],
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "aoss:collection": "cloudwatch-logs-*"
        }
    }
}
]
}

```

Note

之前的角色授予读取账户中所有日志组的权限，使您能够为任何日志账户（包括跨账户日志组）创建控制面板。如果您希望限制对特定日志组的访问，并仅为这些日志组创建控制面板，则可以将该策略中的第一条语句更新为以下内容：

```

{
    "Sid": "CloudWatchLogsAccess",
    "Effect": "Allow",
    "Action": [
        "logs:StartQuery",
        "logs:GetLogGroupFields",

```

```
    "logs:GetQueryResults"  
  ],  
  "Resource": [  
    "arn:aws:logs:us-east-1:123456789012:log-group:myLogGroup:*",  
    "arn:aws:logs:us-east-1:123456789012:log-group:myLogGroup"  
  ]  
}
```

使用筛选条件从日志事件创建指标

您可以通过创建一个或多个指标筛选条件来搜索和筛选进入 CloudWatch Logs 的日志数据。指标筛选条件定义在日志数据发送到 CloudWatch Logs 时要在日志数据中查找的术语和模式。CloudWatch Logs 使用这些指标筛选条件将日志数据转换为 CloudWatch 数值指标，供您生成图表或设置警报。

在通过日志筛选条件创建指标时，还可以选择为该指标分配维度和单位。如果指定单位，请确保在您创建筛选条件时指定正确的单位。稍后更改筛选条件的单位将无效。

如果您已配置 Amazon Organizations 并且正在使用成员账户，则可以使用日志集中化将源账户中的日志数据收集到中央监控账户中。

如果使用集中式日志组，则当创建指标筛选条件时，您可以使用以下系统字段维度：

- `@aws.account` - 此维度表示日志事件来源的 Amazon 账户 ID。
- `@aws.region` - 此维度表示生成日志事件的 Amazon 区域。

这些维度有助于识别日志数据的来源，从而对从集中式日志派生的指标进行更精细的筛选和分析。有关更多信息，请参阅 [跨账户跨区域日志集中化](#)。

如果具有订阅的日志组使用日志转换，则会将筛选模式应用于日志事件的转换版本。有关更多信息，请参阅 [在摄取期间转换日志](#)。

Note

仅标准日志类中的日志组支持指标筛选条件。有关日志类的更多信息，请参阅 [日志类](#)。

在查看这些指标或设置警报时，您可以使用任何类型的 CloudWatch 统计数据，包括百分位数统计数据。

Note

仅当没有任何指标的值为负值时，才支持适用于指标的百分位数统计数据。如果您将指标筛选器设置为可以报告负数，则当它的值有负数时，该指标的百分位数统计数据不可用。有关更多信息，请参阅 [百分位数](#)。

筛选条件不会以回溯方式筛选数据。筛选条件只会发布在创建后发生的事件的指标数据点。测试筛选模式时，筛选结果预览最多会显示前 50 条匹配的日志行以进行验证。如果筛选结果的时间戳早于指标创建时间，则不显示任何日志。

目录

- [概念](#)
- [指标筛选条件的筛选条件模式语法](#)
- [创建指标筛选条件](#)
- [列出指标筛选条件](#)
- [删除指标筛选条件](#)

概念

每个指标筛选条件都由以下关键元素组成：

默认值

在提取日志但未找到匹配日志的时间段中报告给指标筛选条件的值。通过将此项设置为 0，您可以确保在每个这样的时间段中都报告了数据，防止出现存在无匹配数据时间段的“断点”指标。如果在 1 分钟时间段内未提取日志，则不会报告任何值。

如果将维度分配给由指标筛选条件创建的指标，则无法为该指标分配默认值。

维度

维度是进一步定义指标的键值对。您可以将维度分配给通过指标筛选条件创建的指标。由于维度是指标的唯一标识符的一部分，因此每当从日志中提取唯一的名称/值对时，都会创建该指标的一个新变体。

筛选条件模式

一种符号描述，说明 CloudWatch Logs 应如何解释每个日志事件中的数据。例如，日志条目可能包含时间戳、IP 地址、字符串等。您可以使用模式来指定要在日志文件中查找的内容。

指标名称

CloudWatch 指标的名称，受监控的日志信息应该发布到此指标。例如，您可以向名为 ErrorCount 的指标发布信息。

指标命名空间

新 CloudWatch 指标的目标命名空间。

指标值

每次发现匹配日志时发布到指标的数字值。例如，如果您要对特定字词 (如“Error”) 的出现次数进行计数，则每出现一次，该值都将增加“1”。如果要计算传输的字节数，您可以按照在日志事件中找到的实际字节数累加。

指标筛选条件的筛选条件模式语法

Note

指标筛选条件与 CloudWatch Logs Insights 查询有何不同

指标筛选条件与 CloudWatch Logs Insights 查询的不同之处在于，每次找到匹配的日志时，都会将指定的数值添加到指标筛选条件中。有关更多信息，请参阅 [为指标筛选条件配置指标值](#)。

有关如何使用 Amazon CloudWatch Logs Insights 查询语言查询日志组的信息，请参阅 [CloudWatch Logs Insights 语法查询语法](#)。

通用筛选条件模式示例

有关适用于指标筛选条件以及[订阅筛选条件](#)和[筛选日志事件](#)的通用筛选条件模式语法的更多信息，请参阅[指标筛选条件、订阅筛选条件和筛选日志事件的筛选条件模式语法](#)，其中包括以下示例：

- 支持的正则表达式 (regex) 语法
- 匹配非结构化日志事件中的字词
- 匹配 JSON 日志事件中的字词
- 匹配以空格分隔的日志事件中的字词

指标筛选条件能帮助您搜索和筛选进入 CloudWatch Logs 的日志数据，从筛选的日志数据中提取指标观察结果，并将数据点转换为 CloudWatch Logs 指标。您定义在将日志数据发送到 CloudWatch Logs 时要在其中查找的字词和模式。指标筛选条件将分配给日志组，分配给日志组的所有筛选条件都将应用于其日志流。

指标筛选条件与字词匹配时，它会通过指定的数值来增加指标的计数。例如，您可以创建一个指标筛选条件以计算录入事件中单词 ERROR 出现的次数。

您可以为指标分配度量单位和维度。例如，如果您创建了一个指标筛选条件，该筛选条件计算了单词 ERROR 在您的日志事件中出现的次数，则可以指定名为 ErrorCode 的维度，以显示包含单词 ERROR 的日志事件总数，并按报告的错误代码筛选数据。

 Tip

为指标分配度量单位时，请确保指定正确的单位。如果稍后更改单位，则您的更改可能无法生效。有关 CloudWatch 支持的单位的完整列表，请参阅《Amazon CloudWatch API Reference》中的 [MetricDatum](#)。

主题

- [为指标筛选条件配置指标值](#)
- [使用来自 JSON 或以空格分隔的录入事件的指标发布维度](#)
- [使用录入事件中的值来增加指标的值](#)

为指标筛选条件配置指标值

创建指标筛选条件时，您可以定义筛选条件模式并指定指标的值和原定设置值。您可以将指标值设置为数字、命名标识符或数字标识符。如果您没有指定原定设置值，则当指标筛选条件找不到匹配项时，CloudWatch 将不会报告数据。即使该值为 0，也建议您指定原定设置值。设置原定设置值有助于 CloudWatch 更准确地报告数据，并防止 CloudWatch 聚合断点指标。CloudWatch 每分钟聚合和报告指标值。

当您的指标筛选条件在录入事件中找到匹配项时，它会按指标值增加指标的计数。如果指标筛选条件找不到匹配项，则 CloudWatch 会报告该指标的原定设置值。例如，您的日志组每分钟发布两条记录，指标值为 1，原定设置值为 0。如果指标筛选条件在第一分钟内的两个日志记录中均找到了匹配项，则该分钟的指标值为 2。如果指标筛选条件在第二分钟内未找到任何记录中的匹配项，则该分钟的原定设置值为 0。如果将维度分配给指标筛选条件生成的指标，则无法为这些指标指定原定设置值。

您还可以设置指标筛选条件，使用从录入事件中提取的值而不是静态值来增加指标。有关更多信息，请参阅 [使用录入事件中的值来增加指标的值](#)。

使用来自 JSON 或以空格分隔的录入事件的指标发布维度

您可以使用 CloudWatch 控制台或 Amazon CLI 来创建指标筛选条件，该筛选条件使用 JSON 和以空格分隔的录入事件生成的指标来发布维度。维度是名称/值对，仅适用于 JSON 和以空格分隔的筛选条件模式。您可以创建最多三个维度的 JSON 和以空格分隔的指标筛选条件。有关维度的更多信息以及有关如何将维度分配给指标的信息，请参阅以下各部分：

- 《Amazon CloudWatch 用户指南》中的 [维度](#)

- 《Amazon CloudWatch Logs 用户指南》中的[示例：从 Apache 日志中提取字段并分配维度](#)

Important

维度包含收集费用的值，与自定义指标相同。为了防止意外费用，请不要将高基数字段（例如 IPAddress 或 requestId）指定为维度。

从录入事件中提取的指标按自定义指标收费。为了防止意外的高额费用，如果指标筛选条件为在一定时间内指定的维度生成 1000 个不同的名称/值对，则 Amazon 可能会禁用该指标筛选条件。

您可以创建账单告警，通知您预估费用。有关更多信息，请参阅[创建账单告警以监控 Amazon 预估费用](#)。

使用来自 JSON 日志事件的指标发布维度

以下示例包含描述如何在 JSON 指标筛选条件中指定维度的代码段。

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

如果您使用示例 JSON 录入事件测试示例指标筛选条件，则必须在单行中输入示例 JSON 日志。

Example: Metric filter

每当 JSON 录入事件包含属性 `eventType` 和 `"sourceIPAddress"` 时，指标筛选条件都会增加指标。

```
{ $.eventType = "*" && $.sourceIPAddress != 123.123.* }
```

在创建 JSON 指标筛选条件时，您可以将指标筛选条件中的任何属性指定为维度。例如，要设置 `eventType` 为维度，请使用以下内容：

```
"eventType" : $.eventType
```

示例指标包含名为 `"eventType"` 的维度，并且示例录入事件中维度的值为 `"UpdateTrail"`。

使用来自以空格分隔的日志事件的指标发布维度

以下示例包含描述如何在以空格分隔的指标筛选条件中指定维度的代码段。

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]
```

当以空格分隔的录入事件包含筛选条件中指定的任何字段时，指标筛选条件将增加指标。例如，指标筛选条件在以空格分隔的录入事件示例中查找以下字段和值。

```
{
  "$bytes": "1534",
  "$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"
}
```

在创建以空格分隔的指标筛选条件时，您可以将指标筛选条件中的任何字段指定为维度。例如，要设置 `server` 为维度，请使用以下内容：

```
"server" : $server
```

示例指标筛选条件包含名为 `server` 的维度，并且示例录入事件中维度的值为 `"Prod"`。

Example: Match terms with AND (&&) and OR (||)

您可以使用逻辑运算符 AND ("&&") 和 OR ("||") 创建包含条件的以空格分隔的指标筛选条件。以下指标筛选条件会返回日志事件，其中事件中的第一个单词是 ERROR 或 WARN 的任何超字符串。

```
[w1=ERROR || w1=%WARN%, w2]
```

使用录入事件中的值来增加指标的值

您可以创建指标筛选条件，该筛选条件会发布在录入事件中找到的数字值。本部分中的过程使用以下示例指标筛选条件来显示如何将 JSON 录入事件中的数字值发布到指标。

```
{ $.latency = * } metricValue: $.latency
```

创建在录入事件中发布值的指标筛选条件

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在左侧导航窗格中，选择 Logs (日志)，然后选择 Log groups (日志组)。
3. 选择或创建日志组。

有关如何建立日志组的更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[在 CloudWatch Logs 中创建日志组](#)。

4. 选择 Actions (操作)，然后选择 Create metric filter (创建指标筛选条件)。
5. 对于 Filter Pattern (筛选条件模式)，输入 **{ \$.latency = * }**，然后选择 Next (下一步)。
6. 对于 Metric Name (指标名称)，输入 myMetric。
7. 对于 Metric Value (指标值)，输入 **\$.latency**。
8. (可选) 对于 Default Value (原定设置值)，输入 0，然后选择 Next (下一步)。

即使该值为 0，也建议您指定原定设置值。设置原定设置值有助于 CloudWatch 更准确地报告数据，并防止 CloudWatch 聚合断点指标。CloudWatch 每分钟聚合和报告指标值。

9. 选择 Create metric filter (创建指标筛选条件)。

示例指标筛选条件与示例 JSON 录入事件中的字词 "latency" 匹配，并将 50 的数字值发布到指标 myMetric。

```
{
  "latency": 50,
  "requestType": "GET"
}
```

创建指标筛选条件

以下过程和示例介绍如何创建指标筛选条件。

示例

- [创建日志组的指标筛选条件](#)
- [示例：对日志事件进行计数](#)

- [示例：对字词的出现次数进行计数](#)
- [示例：对 HTTP 404 代码进行计数](#)
- [示例：对 HTTP 4xx 代码进行计数](#)
- [示例：从 Apache 日志中提取字段并分配维度](#)

创建日志组的指标筛选条件

要创建日志组的指标筛选条件，请执行下列步骤。在拥有一些数据点之前，该指标将不可见。

使用 CloudWatch 控制台创建指标筛选条件

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在左侧导航窗格中，选择 Logs (日志)，然后选择 Log groups (日志组)。
3. 选择日志组的名称。
4. 选择 Actions，然后选择 Create metric filter (创建指标筛选条件)。
5. 对于 Filter pattern (筛选条件模式)，输入一种筛选条件模式。有关更多信息，请参阅 [指标筛选条件、订阅筛选条件、筛选日志事件和 Live Tail 的筛选条件模式语法](#)。
6. (可选) 如果您使用的是集中式日志组，则可以在筛选条件选择标准下，根据源账户 (@aws.account)、源区域 (@aws.region) 或这两个条件指定筛选条件。
7. (可选) 要测试您的筛选条件模式，请在 Test Pattern (测试模式) 下，输入一个或多个用于测试模式的日志事件。每个日志事件必须格式化为一行。换行符用于分隔 Log event messages (日志事件消息) 框中的日志事件。
8. 选择 Next (下一步)，然后为指标筛选条件输入名称。
9. 在 Metric details (指标详细信息) 下，对于 Metric namespace (指标命名空间)，输入将在其中发布指标的 CloudWatch 命名空间的名称。如果该命名空间尚不存在，请确保选中 Create new (新建)。
10. 对于 Metric name (指标名称)，为新指标输入名称。
11. 对于 Metric value (指标值)，如果您的指标筛选条件正在计算筛选条件中关键字的出现次数，请输入 1。对于包含其中一个关键字的每个日志事件，此操作会以 1 为增量增加该指标。

或者，输入令牌，例如 `$size`。对于包含 size 字段的每个日志事件，此操作会以 size 字段中的数值为增量增加该指标。
12. (可选) 对于 Unit (单位)，选择要分配给相应指标的单位。如果未指定单位，则单位将设置为 None。

13. (可选) 为指标输入多达三个维度的名称和令牌。如果将维度分配给指标筛选条件创建的指标，则无法为这些指标分配默认值。

 Note

维度仅在 JSON 或以空格分隔的指标筛选条件中受支持。

14. 选择 Create metric filter (创建指标筛选条件)。可以从导航窗格找到您创建的指标筛选条件。选择 Logs (日志)，然后选择 Log groups (日志组)。选择您为其创建指标筛选条件的日志组的名称，然后选择 Metric filters (指标筛选条件) 选项卡。

示例：对日志事件进行计数

最简单的日志事件监控就是对发生的日志事件进行计数。您可能想对所有事件进行计数，以创建“检测信号”式监视器，或只是练习创建指标筛选条件。

在以下 CLI 示例中，将名为 MyAppAccessCount 的指标筛选条件应用于日志组 MyApp/access.log，以在 CloudWatch 命名空间 MyNamespace 中创建指标 EventCount。该筛选条件配置为与任何日志事件内容匹配并以“1”为增量增加该指标。

使用 CloudWatch 控制台创建指标筛选条件

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Log groups (日志组)。
3. 选择日志组的名称。
4. 选择 Actions、Create metric filter (创建指标筛选条件)。
5. 将 Filter Pattern (筛选条件模式) 和 Select Log Data to Test (选择要测试的日志数据) 保留为空。
6. 选择 Next (下一步)，然后对于 Filter Name (筛选条件名称)，键入 **EventCount**。
7. 在 Metric Details (指标详细信息) 下，为 Metric Namespace (指标命名空间) 键入 **MyNameSpace**。
8. 对于 Metric Name (指标名称)，键入 **MyAppEventCount**。
9. 确认 Metric Value (指标值) 为 1。这指定对于每个日志事件，计数以 1 累加。
10. 对于 Default Value (默认值)，输入 0，然后选择 Next (下一步)。指定默认值可确保即使在未出现日志事件的时间段内也报告有数据，防止出现有时不存在数据的断点指标。

11. 选择 Create metric filter (创建指标筛选条件)。

使用 Amazon CLI 创建指标筛选条件

在命令提示符处，运行以下命令：

```
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
```

您可以通过发布任何事件数据来测试此新策略。您应该看到有数据点发布到指标 MyAppAccessEventCount。

使用 Amazon CLI 发布事件数据

在命令提示符处，运行以下命令：

```
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"
```

示例：对字词的出现次数进行计数

日志事件经常包含您需要计数的重要消息，可能是关于操作的成功或失败的消息。例如，如果给定操作失败，可能发生错误，且错误记录到日志文件中。您可能需要监控这些日志条目以了解错误发生趋势。

在以下示例中，创建了一个指标筛选条件来监控“Error”这个词。策略已创建并添加到日志组 MyApp/message.log。对于每个包含“Error”的事件，CloudWatch Logs 都会将一个数据点发布到 MyApp/message.log 命名空间中值为“1”的 CloudWatch 自定义指标 ErrorCount。如果没有任何事件包含“Error”这个单词，则发布值 0。当在 CloudWatch 控制台中绘制该数据的图表时，请务必使用总计统计数据。

创建指标筛选条件后，您可以在 CloudWatch 控制台中查看指标。选择要查看的指标时，请选择与日志组名称匹配的指标命名空间。有关更多信息，请参阅[查看可用指标](#)。

使用 CloudWatch 控制台创建指标筛选条件

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Log groups (日志组)。
3. 选择日志组的名称。
4. 选择 Actions (操作)、Create metric filter (创建指标筛选条件)。
5. 对于 Filter pattern (筛选条件模式)，输入 **Error**。

Note

Filter Pattern (筛选条件模式) 中的所有条目都区分大小写。

6. (可选) 要测试您的筛选条件模式，请在 Test Pattern (测试模式) 下，输入一个或多个用于测试模式的日志事件。每个日志事件必须位于一行内，因为换行符用于在 Log event messages (日志事件消息) 框中分隔日志事件。
7. 选择 Next (下一步)，然后在 Assign metric (分配指标) 页上，对于 Filter Name (筛选条件名称)，键入 **MyAppErrorCount**。
8. 在 Metric Details (指标详细信息) 下，为 Metric Namespace (指标命名空间) 键入 MyNamespace。
9. 为 Metric Name (指标名称) 键入 ErrorCount。
10. 确认 Metric Value (指标值) 为 1。这指定对于每个包含“Error”的日志事件，计数以 1 累加。
11. 对于 Default Value (默认值)，键入 0，然后选择 Next (下一步)。
12. 选择 Create metric filter (创建指标筛选条件)。

使用 Amazon CLI 创建指标筛选条件

在命令提示符处，运行以下命令：

```
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
```

您可以通过在消息中发布包含“Error”这个词的事件来测试此新策略。

使用 Amazon CLI 发布事件

在命令提示符处，运行以下命令。注意，模式区分大小写。

```
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"
```

示例：对 HTTP 404 代码进行计数

使用 CloudWatch Logs，您可以监控 Apache 服务器返回 HTTP 404 响应的次数，404 是表示找不到页面的响应代码。您可能需要对此进行监控，从而了解站点来访者找不到所查找资源的频率。假定日志记录是结构化的，每一个日志记录 (站点访问) 都包含以下信息：

- 请求者 IP 地址
- RFC 1413 标识
- 用户名
- Timestamp
- 请求方法以及请求的资源 and 协议
- 对请求的 HTTP 响应代码
- 请求中传输的字节数

这种日志记录的示例如下所示：

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 404 2326
```

您可以指定一个规则，让它尝试匹配其结构反应 HTTP 404 错误的事件，如下例所示：

使用 CloudWatch 控制台创建指标筛选条件

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Log groups (日志组)。
3. 选择 Actions、Create metric filter (创建指标筛选条件)。

4. 对于 Filter Pattern (筛选条件模式) , 键入 **[IP, UserInfo, User, Timestamp, RequestInfo, StatusCode=404, Bytes]**。
5. (可选) 要测试您的筛选条件模式, 请在 Test Pattern (测试模式) 下, 输入一个或多个用于测试模式的日志事件。每个日志事件必须位于一行内, 因为换行符用于在 Log event messages (日志事件消息) 框中分隔日志事件。
6. 选择 Next (下一步) , 然后对于 Filter Name (筛选条件名称) , 键入 HTTP404Errors。
7. 在 Metric Details (指标详细信息) 下, 对于 Metric Namespace (指标命名空间) , 输入 **MyNameSpace**。
8. 对于 Metric Name (指标名称) , 输入 **ApacheNotFoundErrorCode**。
9. 确认 Metric Value (指标值) 为 1。这指定对于每个“404 Error”事件, 计数以 1 累加。
10. 对于 Default Value (默认值) , 输入 0, 然后选择 Next (下一步) 。
11. 选择 Create metric filter (创建指标筛选条件) 。

使用 Amazon CLI 创建指标筛选条件

在命令提示符处, 运行以下命令 :

```
aws logs put-metric-filter \
  --log-group-name MyApp/access.log \
  --filter-name HTTP404Errors \
  --filter-pattern '[ip, id, user, timestamp, request, status_code=404, size]' \
  --metric-transformations \
    metricName=ApacheNotFoundErrorCode,metricNamespace=MyNameSpace,metricValue=1
```

此示例中使用了文字字符, 如左右方括号、双引号和字符串 404。该模式需要整个日志事件消息匹配, 才能考虑监控日志事件。

您可以使用 describe-metric-filters 命令来验证指标筛选条件的创建。应看到类似如下内容的输出 :

```
aws logs describe-metric-filters --log-group-name MyApp/access.log

{
  "metricFilters": [
    {
      "filterName": "HTTP404Errors",
      "metricTransformations": [
        {
          "metricValue": "1",
```

```

        "metricNamespace": "MyNamespace",
        "metricName": "ApacheNotFoundErrorCode"
    }
],
"creationTime": 1399277571078,
"filterPattern": "[ip, id, user, timestamp, request, status_code=404,
size]"
}
]
}

```

现在，您可以手动发布一些事件：

```

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"

```

在发布这些日志事件示例后，您很快可以在 CloudWatch 控制台中检索名为 ApacheNotFoundErrorCode 的指标。

示例：对 HTTP 4xx 代码进行计数

如前面的示例一样，您可能需要监控 Web 服务访问日志和监控 HTTP 响应代码级别。例如，您可能需要监控所有 HTTP 400 级的错误。但是，您可能不想为每一个返回代码指定新的指标筛选条件。

以下示例演示如何创建包括访问日志中所有 400 级别 HTTP 代码响应的指标，该访问日志使用 [示例：对 HTTP 404 代码进行计数](#) 示例中的 Apache 访问日志格式。

使用 CloudWatch 控制台创建指标筛选条件

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Log groups (日志组)。
3. 选择 Apache 服务器的日志组的名称。
4. 选择 Actions、Create metric filter (创建指标筛选条件)。
5. 对于 Filter pattern (筛选条件模式)，输入 **[ip, id, user, timestamp, request, status_code=4*, size]**。

6. (可选) 要测试您的筛选条件模式, 请在 Test Pattern (测试模式) 下, 输入一个或多个用于测试模式的日志事件。每个日志事件必须位于一行内, 因为换行符用于在 Log event messages (日志事件消息) 框中分隔日志事件。
7. 选择 Next (下一步), 然后对于 Filter name (筛选条件名称), 键入 **HTTP4xxErrors**。
8. 在 Metric Details (指标详细信息) 下, 对于 Metric namespace (指标命名空间), 输入 **MyNamespace**。
9. 对于 Metric name (指标名称), 输入 HTTP4xxErrors。
10. 对于 Metric Value (指标值), 输入 1。这指定对于每个包含 4xx 错误的日志事件, 计数以 1 累加。
11. 对于 Default value (默认值), 输入 0, 然后选择 Next (下一步)。
12. 选择 Create metric filter (创建指标筛选条件)。

使用 Amazon CLI 创建指标筛选条件

在命令提示符处, 运行以下命令:

```
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
```

您可以在 put-event 调用中使用以下数据来测试此规则。如果不删除前例中的监控规则, 则会生成两个不同的指标。

```
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
```

示例: 从 Apache 日志中提取字段并分配维度

有时, 使用各个日志事件中的值而不是使用计数作为指标值非常有用。此示例介绍如何创建提取规则, 以创建衡量 Apache Web 服务器传输的字节数的指标。

此示例还展示了如何将维度分配给您正在创建的指标。

使用 CloudWatch 控制台创建指标筛选条件

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Log groups (日志组)。
3. 选择 Apache 服务器的日志组的名称。
4. 选择 Actions、Create metric filter (创建指标筛选条件)。
5. 对于 Filter pattern (筛选条件模式)，输入 **[ip, id, user, timestamp, request, status_code, size]**。
6. (可选) 要测试您的筛选条件模式，请在 Test Pattern (测试模式) 下，输入一个或多个用于测试模式的日志事件。每个日志事件必须位于一行内，因为换行符用于在 Log event messages (日志事件消息) 框中分隔日志事件。
7. 选择 Next (下一步)，然后对于 Filter name (筛选条件名称)，键入 **size**。
8. 在 Metric Details (指标详细信息) 下，对于 Metric namespace (指标命名空间)，输入 **MyNameSpace**。因为这是一个新的命名空间，请确保已选中 Create new (新建)。
9. 对于 Metric name (指标名称)，输入 **BytesTransferred**
10. 对于 Metric value (指标值)，输入 **\$size**。
11. 对于 Unit (单位)，选择 Bytes (字节)。
12. 对于 Dimension Name (维度名称)，键入 **IP**。
13. 对于 Dimension Value (维度值)，键入 **\$ip**，然后选择 Next (下一步)。
14. 选择 Create metric filter (创建指标筛选条件)。

使用 Amazon CLI 创建该指标筛选条件

在命令提示符处，运行以下命令

```
aws logs put-metric-filter \  
--log-group-name MyApp/access.log \  
--filter-name BytesTransferred \  
--filter-pattern '[ip, id, user, timestamp, request, status_code, size]' \  
--metric-transformations \  
metricName=BytesTransferred,metricNamespace=MyNamespace,metricValue='$size'
```

```
aws logs put-metric-filter \  

```

```
--log-group-name MyApp/access.log \
--filter-name BytesTransferred \
--filter-pattern '[ip, id, user, timestamp, request, status_code, size]' \
--metric-transformations \
metricName=BytesTransferred,metricNamespace=MyNamespace,metricValue='$size',unit=Bytes,dimension1=$ip}}'
```

Note

在此命令中，使用此格式指定多个维度。

```
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}'
```

您可以在 put-log-event 调用中使用以下数据来测试此规则。如果不删除前例中的监控规则，则会生成两个不同的指标。

```
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
```

列出指标筛选条件

您可以列出日志组的所有指标筛选条件。

使用 CloudWatch 控制台列出指标筛选条件

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Log groups (日志组)。
3. 在内容窗格的日志组列表中，在 Metric Filters (指标筛选条件) 列中选择筛选条件数目。

Log Groups > Filters for (日志组 > 筛选条件) 屏幕将列出与该日志组关联的所有指标筛选条件。

使用 Amazon CLI 列出指标筛选条件

在命令提示符处，运行以下命令：

```
aws logs describe-metric-filters --log-group-name MyApp/access.log
```

下面是示例输出：

```
{
  "metricFilters": [
    {
      "filterName": "HTTP404Errors",
      "metricTransformations": [
        {
          "metricValue": "1",
          "metricNamespace": "MyNamespace",
          "metricName": "ApacheNotFoundErrorCode"
        }
      ],
      "creationTime": 1399277571078,
      "filterPattern": "[ip, id, user, timestamp, request, status_code=404,
size]"
    }
  ]
}
```

删除指标筛选条件

策略由其名称和它所属的日志组确定。

使用 CloudWatch 控制台删除指标筛选条件

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Log groups (日志组)。
3. 在内容窗格的 Metric Filter (指标筛选条件) 列中，为日志组选择指标筛选条件的数量。
4. 在 Metric Filters (指标筛选条件) 屏幕中，选中要删除的筛选条件名称右侧的复选框。然后选择 Delete (删除)。

5. 当系统提示进行确认时，选择 Delete (删除)。

使用 Amazon CLI 删除指标筛选条件

在命令提示符处，运行以下命令：

```
aws logs delete-metric-filter --log-group-name MyApp/access.log \  
--filter-name MyFilterName
```

使用订阅实时处理日志数据

您可以使用订阅从 CloudWatch Logs 中访问日志事件的实时源，并将其传输到其他服务（如 Amazon Kinesis 流、Amazon Data Firehose 流或 Amazon Lambda）以进行自定义处理、分析或加载到其他系统中。当将录入事件发送到接收它们的服务时，它们会经过 base64 编码并使用 gzip 格式进行压缩。

您还可以使用 CloudWatch Logs 集中化将来自多个账户和区域的日志数据复制到中心位置。有关更多信息，请参阅 [跨账户跨区域日志集中化](#)。

要开始订阅日志事件，请创建用于接收事件的接收资源，例如 Kinesis Data Streams 流。订阅筛选条件定义了筛选条件模式（用于筛选传输到您的 Amazon 资源的日志事件），以及有关匹配日志事件发送目的地的信息。日志事件在摄取后不久（通常不到三分钟）就会发送到接收资源。

Note

如果具有订阅的日志组使用日志转换，则会将筛选模式与日志事件的转换版本进行比较。有关更多信息，请参阅 [在摄取期间转换日志](#)。

您可以在账户级别和日志组级别创建订阅。每个账户对于每个区域可以有一个账户级订阅筛选条件。每个日志组最多只能有两个与其关联的订阅筛选条件。

Note

如果目标服务返回一个可重试错误，如限制异常或可重试服务异常（如 HTTP 5xx），则 CloudWatch Logs 将继续重试传输长达 24 小时。如果错误是不可重试错误（例如 AccessDeniedException 或 ResourceNotFoundException），则 CloudWatch Logs 不会尝试重试传输。在这些情况下，订阅筛选条件将被禁用长达 10 分钟，然后 CloudWatch Logs 会重试将日志发送到目的地。在此禁用期间，将跳过日志。

CloudWatch Logs 还会生成有关将日志事件转发到订阅的 CloudWatch 指标。有关更多信息，请参阅 [使用 CloudWatch 指标进行监控](#)。

您还可使用 CloudWatch Logs 订阅将日志数据近乎实时地流式传输到 Amazon OpenSearch Service 集群。有关更多信息，请参阅 [将 CloudWatch Logs 数据流式传输到 Amazon OpenSearch Service](#)。

仅标准日志类中的日志组支持订阅。有关日志类的更多信息，请参阅 [日志类](#)。

Note

订阅筛选条件可能会批量记录事件以优化传输并减少对目的地的调用量。不保证使用批处理，但会尽可能地使用。

目录

- [概念](#)
- [日志组级别订阅筛选条件](#)
- [账户级订阅筛选条件](#)
- [跨账户跨区域订阅](#)
- [混淆代理问题防范](#)
- [防止日志递归](#)

概念

每个订阅筛选条件都由以下关键元素组成：

筛选条件模式

一种符号描述，说明 CloudWatch Logs 应如何解释每个日志事件中的数据以及可限制传输到目标 Amazon 资源的内容的筛选表达式。有关筛选条件模式语法的更多信息，请参阅 [指标筛选条件、订阅筛选条件、筛选日志事件和 Live Tail 的筛选条件模式语法](#)。

目标 ARN

要用作订阅源目的地的 Kinesis Data Streams 流、Firehose 流或 Lambda 函数的 Amazon 资源名称 (ARN)。

角色 ARN

一个 IAM 角色，用于向 CloudWatch Logs 授予将数据放入所选目标所需的必要权限。此角色不适用于 Lambda 目标，因为 CloudWatch Logs 可以通过 Lambda 函数本身的访问控制设置获得必要的权限。

分配

当目标是 Amazon Kinesis Data Streams 流中的流时，用于将日志数据分配到目标的方法。默认情况下，日志数据按日志流进行分组。为了实现更均匀的分配，您可以对日志数据进行随机分组。

对于日志组级别的订阅，还包括下面的关键元素：

日志组名称

要将订阅筛选条件关联到的日志组。上传到此日志组的所有日志事件都受订阅筛选条件的约束，与该筛选条件匹配的日志事件将被传输到接收匹配日志事件的目标服务。

对于账户级别的订阅，还包括下面的关键元素：

选择标准

用于选择哪些日志组应用了账户级别订阅筛选条件的标准。如果不指定此元素，则账户级别订阅筛选条件将应用于账户中的所有日志组。此字段用于防止无限日志循环。有关无限日志循环问题的更多信息，请参阅日志 [防止日志递归](#)。

选择标准的大小限制为 25 KB。

对于集中式日志组，还包括下面的关键元素。这些元素可用作字段选择标准，以帮助识别日志数据的来源，从而对从集中式日志派生的指标进行更精细的筛选和分析。

@aws.account

此字段标识日志事件来源的 Amazon 账户 ID。

@aws.region

此字段标识生成日志事件的 Amazon 区域。

日志组级别订阅筛选条件

您可以将订阅筛选条件与 Amazon Kinesis Data Streams、Amazon Lambda、Amazon Data Firehose 或 Amazon OpenSearch Service 结合使用。通过订阅筛选条件发送到服务的日志将经过 base64 编码并使用 gzip 格式进行压缩。如果您将集中式日志与 Amazon Organizations 结合使用，则可以选择发出 @aws.account 和 @aws.region 系统字段，以识别哪些数据来自您组织中的哪些账户和区域。此部分提供了一些示例，您可以按照这些示例创建 CloudWatch Logs 订阅筛选条件来将日志数据发送到 Firehose、Lambda、Kinesis Data Streams 和 OpenSearch Service。

Note

如果要搜索日志数据，请参阅 [Filter and pattern syntax](#)。

示例

- [示例 1：Kinesis Data Streams 订阅筛选条件](#)
- [示例 2：Amazon Lambda 订阅筛选条件](#)
- [示例 3：Amazon Data Firehose 订阅筛选条件](#)
- [示例 4：Amazon OpenSearch Service 订阅筛选条件](#)

示例 1：Kinesis Data Streams 订阅筛选条件

以下示例将订阅筛选条件与包含 Amazon CloudTrail 事件的日志组关联。订阅筛选条件将每个由 "Root" Amazon 凭证记录的活动传递至 Kinesis Data Streams 流中名为 "RootAccess" 的流。有关如何将 Amazon CloudTrail 事件发送到 CloudWatch Logs 的更多信息，请参阅 Amazon CloudTrail 用户指南中的 [将 CloudTrail 事件发送到 CloudWatch Logs](#)。

Note

在创建流之前，请计算将生成日志数据的卷。请确保创建具有足够分片的流来处理此卷。如果流没有足够的分片，日志流将受限制。有关流卷限制的更多信息，请参阅[限额和限制](#)。受限的交付项最多可在 24 小时内重试。24 小时后，失败的交付项将被丢弃。要减轻节流的风险，您可以执行以下步骤：

- 使用 [PutSubscriptionFilter](#) 或 [put-subscription-filter](#) 创建订阅筛选条件时，请为 `distribution` 指定 `random`。默认情况下，流筛选条件分配按日志流进行，这可能会导致节流。
- 使用 CloudWatch 指标监控您的流。这可以帮助您识别任何节流并相应地调整配置。例如，`DeliveryThrottling` 指标可用于跟踪将数据转发到订阅目标时，CloudWatch Logs 因为其而被限制的日志事件的数量。有关监控的更多信息，请参阅[使用 CloudWatch 指标进行监控](#)。
- 为您在 Kinesis Data Streams 中的流使用按需容量模式。按需模式会随着工作负载的增加或减少立即进行调整。有关按需容量模式的更多信息，请参阅[按需模式](#)。

- 限制您的 CloudWatch 订阅筛选模式，使其与 Kinesis Data Streams 中的流容量相匹配。如果您向流发送的数据过多，则可能需要减小筛选器大小或调整筛选条件。

为 Kinesis Data Streams 创建订阅筛选条件

1. 使用以下命令创建目标流：

```
$ C:\> aws kinesis create-stream --stream-name "RootAccess" --shard-count 1
```

2. 请耐心等待，直到流变为活动状态（这可能需要一两分钟）。您可使用以下 Kinesis Data Streams [describe-stream](#) 命令检查 StreamDescription.StreamStatus 属性。此外，请记住 StreamDescription.StreamARN 值，因为后面的步骤中将会用到它：

```
aws kinesis describe-stream --stream-name "RootAccess"
```

下面是示例输出：

```
{
  "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. 创建 IAM 角色，该角色将向 CloudWatch Logs 授予将数据放入流的权限。首先，您需要在文件 (例如 `~/TrustPolicyForCWL-Kinesis.json`) 中创建信任策略。使用文本编辑器创建此策略。请勿使用 IAM 控制台来创建策略。

此策略包括 `aws:SourceArn` 全局条件上下文密钥，有助于避免出现混淆代理安全问题。有关更多信息，请参阅 [混淆代理问题防范](#)。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringLike": { "aws:SourceArn": "arn:aws:logs:region:123456789012:*" }
    }
  }
}
```

4. 使用 `create-role` 命令创建 IAM 角色，并指定信任策略文件。请记住返回的 `RoleArn` 值，因为后面的步骤中将会用到它：

```
aws iam create-role --role-name CWLtoKinesisRole --assume-role-policy-document
file:///~/TrustPolicyForCWL-Kinesis.json
```

下面是输出的一个示例。

```
{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "logs.amazonaws.com"
        },
        "Condition": {
          "StringLike": {
            "aws:SourceArn": { "arn:aws:logs:region:123456789012:*" }
          }
        }
      }
    }
  },
```



```

    "RoleId": "AA0IIAH450GAB4HC5F431",
    "CreateDate": "2015-05-29T13:46:29.431Z",
    "RoleName": "CWLtoKinesisRole",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisRole"
  }
}

```

5. 创建权限策略以定义可对您的账户执行的 CloudWatch Logs 操作。首先，您将在文件 (例如 ~/PermissionsForCWL-Kinesis.json) 中创建权限策略。使用文本编辑器创建此策略。请勿使用 IAM 控制台来创建策略。

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:123456789012:stream/RootAccess"
    }
  ]
}

```

6. 使用以下 [put-role-policy](#) 命令将权限策略与角色关联：

```

aws iam put-role-policy --role-name CWLtoKinesisRole --policy-name Permissions-
Policy-For-CWL --policy-document file://~/PermissionsForCWL-Kinesis.json

```

7. 在流变为活动状态并且您已创建 IAM 角色后，您便可以创建 CloudWatch Logs 订阅筛选条件。订阅筛选条件将立即让实时日志数据开始从所选日志组流动到您的流：

```

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. 设置订阅筛选条件后，CloudWatch Logs 会将与筛选条件模式匹配的所有传入事件日志转发到您的流。您可以抓取 Kinesis Data Streams 分片迭代器，并使用 Kinesis Data Streams get-records 命令提取一些 Kinesis Data Streams 记录以验证是否发生该过程：

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

```
{
  "ShardIterator":
  "AAAAAAAAAAFGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRgB9v4scv+3vaq+f+0IK8zM5My8ID
+g6rMo7UKWeI4+IWiK20Sh0uP"
}
```

```
aws kinesis get-records --limit 10 --shard-iterator "AAAAAAAAAAFGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRgB9v4scv+3vaq+f+0IK8zM5My8ID
+g6rMo7UKWeI4+IWiK20Sh0uP"
```

请注意，您可能需要在 Kinesis Data Streams 开始返回数据之前调用几次此命令。

您将看到包含一组记录的响应。Kinesis Data Streams 记录中的数据属性采用 Base64 编码并使用 gzip 格式进行压缩。您可使用以下 Unix 命令检查命令行中的原始数据：

```
echo -n "<Content of Data>" | base64 -d | zcat
```

Base64 解码和解压缩数据被格式化为 JSON 并具有以下结构：

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

```

上述数据架构中的关键元素如下：

owner

原始生成日志数据的Amazon账户 ID。

logGroup

原始日志数据的日志组名称。

logStream

原始日志数据的日志流名称。

subscriptionFilters

与原始日志数据匹配的订阅筛选条件名称的列表。

messageType

数据消息将使用“DATA_MESSAGE”类型。有时候，CloudWatch Logs 可能发出类型为“CONTROL_MESSAGE”的 Kinesis Data Streams 记录，这主要是为了检查目标是否可访问。

logEvents

表示为一组日志事件记录的实际日志数据。“id”属性是每个日志事件的唯一标识符。

示例 2：Amazon Lambda 订阅筛选条件

此示例将创建一个 CloudWatch Logs 订阅筛选条件，用于将日志数据发送到您的 Amazon Lambda 函数。

Note

在创建 Lambda 函数之前，请计算将生成的日志数据量。请确保创建一个处理此卷的函数。如果函数没有足够的卷，日志流将受限制。有关 Lambda 限制的更多信息，请参阅 [Amazon Lambda 限制](#)。

为 Lambda 创建订阅筛选条件

1. 创建 Amazon Lambda 函数。

确保您已设置 Lambda 执行角色。有关更多信息，请参阅 Amazon Lambda 开发人员指南中的 [步骤 2.2：创建 IAM 角色（执行角色）](#)。

2. 打开文本编辑器，并用以下内容创建名为 helloWorld.js 的文件：

```
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. 压缩 helloWorld.js 文件，并用名称 helloWorld.zip 保存它。

4. 使用以下命令，其中的角色是您在第一步中设置的 Lambda 执行角色：

```
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. 授予 CloudWatch Logs 执行您的函数的权限。使用以下命令，将占位符账户替换为您自己的账户，将占位符日志组替换为要处理的日志组：

```
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. 使用以下命令创建订阅筛选条件，将占位符账户替换为您自己的账户，将占位符日志组替换为要处理的日志组：

```
aws logs put-subscription-filter \
  --log-group-name myLogGroup \
  --filter-name demo \
  --filter-pattern "" \
  --destination-arn arn:aws:lambda:region:123456789123:function:helloworld
```

7. (可选) 使用样本日志事件进行测试。在出现命令提示符时，运行以下命令，以将一条简单的日志消息放入已订阅的流中。

要查看 Lambda 函数的输出，请导航至 Lambda 函数，在此处可以查看 `/aws/lambda/helloworld` 中的输出：

```
aws logs put-log-events --log-group-name myLogGroup --log-stream-name stream1 --
log-events "[{\"timestamp\":<CURRENT TIMESTAMP MILLIS> , \"message\": \"Simple
Lambda Test\"}]"
```

您将看到包含一组 Lambda 的响应。Lambda 记录中的数据属性采用 base64 编码并使用 gzip 格式进行压缩。Lambda 接收的实际负载采用以下格式：`{ "awslogs": { "data": "BASE64ENCODED_GZIP_COMPRESSED_DATA" } }`。您可以使用以下 Unix 命令检查命令行中的原始数据：

```
echo -n "<BASE64ENCODED_GZIP_COMPRESSED_DATA>" | base64 -d | zcat
```

Base64 解码和解压缩数据被格式化为 JSON 并具有以下结构：

```
{
  "owner": "123456789012",
  "logGroup": "CloudTrail",
  "logStream": "123456789012_CloudTrail_us-east-1",
  "subscriptionFilters": [
    "Destination"
  ],
  "messageType": "DATA_MESSAGE",
  "logEvents": [
    {
      "id": "31953106606966983378809025079804211143289615424298221568",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\n\"Root\"}}",
    },
    {
      "id": "31953106606966983378809025079804211143289615424298221569",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\n\"Root\"}}",
    },
    {
      "id": "31953106606966983378809025079804211143289615424298221570",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\n\"Root\"}}",
    }
  ]
}
```

上述数据架构中的关键元素如下：

owner

原始生成日志数据的Amazon账户 ID。

logGroup

原始日志数据的日志组名称。

logStream

原始日志数据的日志流名称。

subscriptionFilters

与原始日志数据匹配的订阅筛选条件名称的列表。

messageType

数据消息将使用“DATA_MESSAGE”类型。有时候，CloudWatch Logs 可能发出类型为“CONTROL_MESSAGE”的 Lambda 记录，这主要是为了检查目标是否可访问。

logEvents

表示为一组日志事件记录的实际日志数据。“id”属性是每个日志事件的唯一标识符。

示例 3：Amazon Data Firehose 订阅筛选条件

此示例将创建一个 CloudWatch Logs 订阅，用于将与所定义筛选条件匹配的所有传入日志事件发送到您的 Amazon Data Firehose 传输流。已经使用 gzip 6 级压缩对从 CloudWatch Logs 发送到 Amazon Data Firehose 的数据进行压缩，因此无需在 Firehose 传输流中使用压缩。然后，可以使用 Firehose 中的解压缩功能自动解压缩日志。有关更多信息，请参阅[向 Firehose 发送 CloudWatch Logs](#)。

Note

在创建 Firehose 流之前，请计算将生成的日志数据量。请确保创建一个处理此数据量的 Firehose 流。如果流无法处理卷，则日志流将受限制。有关 Firehose 流数据量限制的更多信息，请参阅[Amazon Data Firehose Data Limits](#)。

为 Firehose 创建订阅筛选条件

1. 创建 Amazon Simple Storage Service (Amazon S3) 存储桶 我们建议您使用专为 CloudWatch Logs 创建的存储桶。但是，如果要使用现有存储桶，请跳至第 2 步。

运行以下命令，将占位符区域替换为您想使用的区域：

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket2 --create-bucket-configuration  
LocationConstraint=region
```

下面是示例输出：

```
{
```

```
"Location": "/amzn-s3-demo-bucket2"
}
```

2. 创建 IAM 角色，该角色将授予 Amazon Data Firehose 将数据放入您的 Amazon S3 存储桶的权限。

有关更多信息，请参阅《Amazon Data Firehose 开发人员指南》中的 [Controlling Access with Amazon Data Firehose](#)。

首先，使用文本编辑器在文件 `~/TrustPolicyForFirehose.json` 中创建一个信任策略，具体如下所示：

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "firehose.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

3. 使用 `create-role` 命令创建 IAM 角色，并指定信任策略文件。请记下返回的 `Role.Arn` 值，因为后面的步骤中将会用到它：

```
aws iam create-role \
  --role-name FirehosetoS3Role \
  --assume-role-policy-document file://~/TrustPolicyForFirehose.json
```

```
{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "firehose.amazonaws.com"
        }
      }
    },
    "RoleId": "AA0IIAH450GAB4HC5F431",
    "CreateDate": "2015-05-29T13:46:29.431Z",
    "RoleName": "FirehosetoS3Role",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:role/FirehosetoS3Role"
```



```
}
}
```

4. 创建权限策略以定义 Firehose 可对您的账户执行的操作。首先，使用文本编辑器在文件 `~/PermissionsForFirehose.json` 中创建权限策略：

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:PutObject" ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket2",
        "arn:aws:s3:::amzn-s3-demo-bucket2/*" ]
    }
  ]
}
```

5. 使用以下 `put-role-policy` 命令将权限策略与角色关联：

```
aws iam put-role-policy --role-name FirehoseToS3Role --policy-name Permissions-
Policy-For-Firehose --policy-document file:///~/PermissionsForFirehose.json
```

6. 如下所示，创建目标 Firehose 传输流，将 RoleARN 和 BucketARN 的占位符值替换为您创建的角色和存储桶 ARN：

```
aws firehose create-delivery-stream \
  --delivery-stream-name 'my-delivery-stream' \
  --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::123456789012:role/FirehoseToS3Role", "BucketARN":
  "arn:aws:s3:::amzn-s3-demo-bucket2"}'
```

请注意，Firehose 对所传输的 Amazon S3 对象自动使用 YYYY/MM/DD/HH UTC 时间格式的前缀。您可以指定要添加在该时间格式前缀前面的额外前缀。如果前缀以正斜杠 (/) 结束，则在 Amazon S3 存储桶中显示为文件夹。

7. 请耐心等待，直到流变为活动状态 (这可能需要几分钟时间)。您可以使用 `Firehose describe-delivery-stream` 命令检查 `DeliveryStreamDescription.DeliveryStreamStatus` 属性。此外，请记下 `DeliveryStreamDescription.DeliveryStreamARN` 值，因为后面的步骤中将会用到它：

```
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-0000000000001",
        "S3DestinationDescription": {
          "CompressionFormat": "UNCOMPRESSED",
          "EncryptionConfiguration": {
            "NoEncryptionConfig": "NoEncryption"
          },
          "RoleARN": "delivery-stream-role",
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket2",
          "BufferingHints": {
            "IntervalInSeconds": 300,
            "SizeInMBs": 5
          }
        }
      }
    ]
  }
}
```

8. 创建 IAM 角色，该角色将授予 CloudWatch Logs 将数据放入 Firehose 传输流的权限。首先，使用文本编辑器在文件 `~/TrustPolicyForCWL.json` 中创建信任策略：

此策略包括 `aws:SourceArn` 全局条件上下文密钥，有助于避免出现混淆代理安全问题。有关更多信息，请参阅 [混淆代理问题防范](#)。

```
{
  "Statement": {
```

```

    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringLike": {
        "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
      }
    }
  }
}

```

9. 使用 `create-role` 命令创建 IAM 角色，并指定信任策略文件。请记下返回的 `Role.Arn` 值，因为后面的步骤中将会用到它：

```

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. 创建权限策略以定义可对您的账户执行的 CloudWatch Logs 操作。首先，使用文本编辑器创建权限策略文件 (例如 `~/PermissionsForCWL.json`)：

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["firehose:PutRecord"],
      "Resource": [
        "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-
name"]
      }
    ]
  }
}
```

11. 使用 `put-role-policy` 命令将权限策略与角色关联：

```
aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-
name Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json
```

12. 在 Amazon Data Firehose 传输流进入活动状态并且您已创建 IAM 角色后，您便可以创建 CloudWatch Logs 订阅筛选条件。订阅筛选条件将立即启动从所选日志组到您的 Amazon Data Firehose 传输流的实时日志数据流动：

```
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. 当您设置订阅筛选条件后，CloudWatch Logs 将与筛选条件模式匹配的所有传入事件日志转发到您的 Amazon Data Firehose 传输流。根据 Amazon Data Firehose 传输流上设置的时间缓冲间隔，数据将开始显示在 Amazon S3 中。经过足够的时间后，您可以通过检查您的 Amazon S3 存储桶验证您的数据。

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket2' --prefix 'firehose/'
{
  "Contents": [
    {
      "LastModified": "2015-10-29T00:01:25.000Z",
```

```

      "ETag": "\"a14589f8897f4089d3264d9e2d1f1610\"",
      "StorageClass": "STANDARD",
      "Key": "firehose/2015/10/29/00/my-delivery-stream-2015-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250",
      "Owner": {
        "DisplayName": "cloudwatch-logs",
        "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b5"
      },
      "Size": 593
    },
    {
      "LastModified": "2015-10-29T00:35:41.000Z",
      "ETag": "\"a7035b65872bb2161388ffb63dd1aec5\"",
      "StorageClass": "STANDARD",
      "Key": "firehose/2015/10/29/00/my-delivery-stream-2015-10-29-00-35-40-7cc92023-7e66-49bc-9fd4-fc9819cc8ed3",
      "Owner": {
        "DisplayName": "cloudwatch-logs",
        "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b6"
      },
      "Size": 5752
    }
  ]
}

```

```

aws s3api get-object --bucket 'amzn-s3-demo-bucket2' --key 'firehose/2015/10/29/00/my-delivery-stream-2015-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250'
testfile.gz

{
  "AcceptRanges": "bytes",
  "ContentType": "application/octet-stream",
  "LastModified": "Thu, 29 Oct 2015 00:07:06 GMT",
  "ContentLength": 593,
  "Metadata": {}
}

```

Amazon S3 对象中的数据以 gzip 格式压缩。您可以使用以下 Unix 命令检查命令行中的原始数据：

```
zcat testfile.gz
```

示例 4：Amazon OpenSearch Service 订阅筛选条件

此示例将创建一个 CloudWatch Logs 订阅，用于将与所定义筛选条件匹配的传入日志事件发送到您的 OpenSearch Service 域。

为 OpenSearch Service 创建订阅筛选条件

1. 创建 OpenSearch Service 域。有关更多信息，请参阅[创建 OpenSearch Service 域](#)
2. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
3. 在导航窗格中，选择 Log groups (日志组)。
4. 选择日志组的名称。
5. 选择 Actions (操作)、Subscription filters (订阅筛选器)、Create Amazon OpenSearch Service subscription filter (创建 Amazon OpenSearch Service 订阅筛选器)。
6. 选择是要流式传输到此账户还是其他账户中的集群。
 - 如果选择此账户，请选择您在步骤 1 中创建的域。
 - 如果选择其他账户，请输入该域的 ARN 和端点。
7. 如果您选择了其他账户，请提供域 ARN 和端点。
8. 对于 Amazon OpenSearch Service 集群，选择要传送日志组数据的集群的名称
9. 选择一种日志格式。
10. 对于订阅筛选条件模式，输入要在您的日志事件中查找的字词或模式。这确保只将您需要的数据发送到 OpenSearch Service 集群。有关更多信息，请参阅[指标筛选条件的筛选条件模式语法](#)。
11. (可选) 对于 Select log data to test (选择要测试的日志数据)，请选择一个日志流，然后选择 Test pattern (测试模式)，以确认搜索筛选器是否会返回您期望的结果。
12. 选择 Start streaming (开始流式传输)。

账户级订阅筛选条件

Important

订阅筛选条件可能会出现无限递归循环的风险，如果不加以解决，可能会导致提取计费大幅增加。为了降低这种风险，我们建议您在账户级订阅筛选条件中使用选择标准来排除从属于订阅交付工作流一部分的资源中提取日志数据的日志组。有关此问题以及确定要排除哪些日志组的更多信息，请参阅[防止日志递归](#)。

您可以设置一个账户级别的订阅策略，其包括账户中的一部分日志组。账户订阅策略可以与 Amazon Kinesis Data Streams、Amazon Lambda 或 Amazon Data Firehose 配合使用。通过账户级订阅策略发送到服务的日志采用 base64 编码并使用 gzip 格式进行压缩。本节提供了一些示例，您可以按照这些示例为 Kinesis Data Streams、Lambda 和 Firehose 创建账户级订阅。

Note

要查看您账户中所有订阅筛选策略的列表，请使用 `describe-account-policies` 命令并将 `--policy-type` 参数的值设置为 `SUBSCRIPTION_FILTER_POLICY`。有关更多信息，请参阅 [describe-account-policies](#)。

示例

- [示例 1：Kinesis Data Streams 订阅筛选条件](#)
- [示例 2：Amazon Lambda 订阅筛选条件](#)
- [示例 3：Amazon Data Firehose 订阅筛选条件](#)

示例 1：Kinesis Data Streams 订阅筛选条件

在创建用于账户级订阅策略的 Kinesis Data Streams 数据流之前，请计算将生成的日志数据量。请确保创建具有足够分片的流来处理此卷。如果某个流没有足够的分片，则会受到限制。有关流数据量限制的更多信息，请参阅 Kinesis Data Streams 文档中的 [Quotas and Limits](#)。

Warning

由于多个日志组的日志事件被转发到目的地，因此存在节流的风险。受限的交付项最多可在 24 小时内重试。24 小时后，失败的交付项将被丢弃。

要减轻节流的风险，您可以执行以下步骤：

- 使用 CloudWatch 指标监控您的 Kinesis Data Streams 流。这可以帮助您识别节流并相应地调整配置。例如，`DeliveryThrottling` 指标跟踪将数据转发到订阅目标时，CloudWatch Logs 因为其而被限制的日志事件的数量。有关更多信息，请参阅 [使用 CloudWatch 指标进行监控](#)。
- 为您在 Kinesis Data Streams 中的流使用按需容量模式。按需模式会随着工作负载的增加或减少立即进行调整。有关更多信息，请参阅 [On-demand mode](#)。

- 限制您的 CloudWatch Logs 订阅筛选模式，使其与 Kinesis Data Streams 中的流容量相匹配。如果您向流发送的数据过多，则可能需要减小筛选器大小或调整筛选条件。

以下示例使用账户级订阅策略将所有日志事件转发到 Kinesis Data Streams 中的流。筛选条件模式将任何日志事件与文本 Test 进行匹配，并将其转发到 Kinesis Data Streams 中的流。

为 Kinesis Data Streams 创建账户级订阅策略

1. 使用以下命令创建目标流：

```
$ C:\> aws kinesis create-stream --stream-name "TestStream" --shard-count 1
```

2. 等待几分钟让流变为活动状态。您可以使用 [describe-stream](#) 命令检查 StreamDescription.StreamStatus 属性来验证该流是否处于活动状态。

```
aws kinesis describe-stream --stream-name "TestStream"
```

下面是示例输出：

```
{
  "StreamDescription": {
    "StreamStatus": "ACTIVE",
    "StreamName": "TestStream",
    "StreamARN": "arn:aws:kinesis:region:123456789012:stream/TestStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "EXAMPLE8463463374607431768211455",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
            "EXAMPLE688818456679503831981458784591352702181572610"
        }
      }
    ]
  }
}
```


3. 创建 IAM 角色，该角色将向 CloudWatch Logs 授予将数据放入流的权限。首先，您需要在文件 (例如 `~/TrustPolicyForCWL-Kinesis.json`) 中创建信任策略。使用文本编辑器创建此策略。

此策略包括 `aws:SourceArn` 全局条件上下文密钥，有助于避免出现混淆代理安全问题。有关更多信息，请参阅 [混淆代理问题防范](#)。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringLike": { "aws:SourceArn": "arn:aws:logs:region:123456789012:*" }
    }
  }
}
```

4. 使用 `create-role` 命令创建 IAM 角色，并指定信任策略文件。请记住返回的 `Role.Arn` 值，因为后面的步骤中将会用到它：

```
aws iam create-role --role-name CWLtoKinesisRole --assume-role-policy-document
file://~/TrustPolicyForCWL-Kinesis.json
```

下面是输出的一个示例。

```
{
  "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. 创建权限策略以定义可对您的账户执行的 CloudWatch Logs 操作。首先，您将在文件 (例如 ~/PermissionsForCWL-Kinesis.json) 中创建权限策略。使用文本编辑器创建此策略。不要使用 IAM 控制台来创建它。

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:123456789012:stream/TestStream"
    }
  ]
}

```

6. 使用以下 [put-role-policy](#) 命令将权限策略与角色关联：

```

aws iam put-role-policy --role-name CWLtoKinesisRole --policy-name Permissions-
Policy-For-CWL --policy-document file://~/PermissionsForCWL-Kinesis.json

```

7. 在流变为活动状态并且您已创建 IAM 角色后，便可以创建 CloudWatch Logs 订阅筛选策略。该策略会立即启动实时日志数据流。在此示例中，所有包含字符串 ERROR 的日志事件都将流式传输，但名为 LogGroupToExclude1 和 LogGroupToExclude2 的日志组中的日志事件除外。

```

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. 设置订阅筛选条件后，CloudWatch Logs 会将与筛选条件模式和选择标准匹配的所有传入事件日志转发到您的流。

`selection-criteria` 字段是可选的，但对于从订阅筛选条件中排除可能导致无限日志递归的日志组非常重要。有关此问题以及确定要排除哪些日志组的更多信息，请参阅 [防止日志递归](#)。目前，NOT IN 是 `selection-criteria` 唯一支持的运算符。

您可以使用 Kinesis Data Streams 分片迭代器并使用 Kinesis Data Streams `get-records` 命令获取一些 Kinesis Data Streams 记录来验证日志事件流：

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

```
{
  "ShardIterator":
  "AAAAAAAAAAAFGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+0IK8zM5My8ID
+g6rMo7UKWeI4+IWIK20Sh0uP"
}
```

```
aws kinesis get-records --limit 10 --shard-iterator "AAAAAAAAAAAFGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+0IK8zM5My8ID
+g6rMo7UKWeI4+IWIK20Sh0uP"
```

在 Kinesis Data Streams 开始返回数据之前，您可能需要使用此命令几次。

您将看到包含一组记录的响应。Kinesis Data Streams 记录中的数据属性采用 Base64 编码并使用 gzip 格式进行压缩。您可使用以下 Unix 命令检查命令行中的原始数据：

```
echo -n "<Content of Data>" | base64 -d | zcat
```

Base64 解码和解压缩数据被格式化为 JSON 并具有以下结构：

```
{
  "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"
}

```

数据架构中的关键元素如下：

messageType

数据消息将使用“DATA_MESSAGE”类型。有时候，CloudWatch Logs 可能发出类型为“CONTROL_MESSAGE”的 Kinesis Data Streams 记录，这主要是为了检查目标是否可访问。

owner

原始生成日志数据的 Amazon 账户 ID。

logGroup

原始日志数据的日志组名称。

logStream

原始日志数据的日志流名称。

subscriptionFilters

与原始日志数据匹配的订阅筛选条件名称的列表。

logEvents

表示为一组日志事件记录的实际日志数据。“id”属性是每个日志事件的唯一标识符。

policyLevel

强制实施策略的级别。“ACCOUNT_LEVEL_POLICY”是账户级订阅筛选策略的 policyLevel。

示例 2：Amazon Lambda 订阅筛选条件

此示例将创建一个 CloudWatch Logs 账户级订阅筛选策略，用于将日志数据发送到您的 Amazon Lambda 函数。

Warning

在创建 Lambda 函数之前，请计算将生成的日志数据量。请确保创建一个处理此卷的函数。如果该函数无法处理该量，则日志流将受限制。由于所有日志组或账户的部分日志组的日志事件都被转发到目标，因此存在节流风险。有关 Lambda 限制的更多信息，请参阅 [Amazon Lambda 限制](#)。

为 Lambda 创建账户级订阅筛选策略

1. 创建 Amazon Lambda 函数。

确保您已设置 Lambda 执行角色。有关更多信息，请参阅 Amazon Lambda 开发人员指南中的 [步骤 2.2：创建 IAM 角色 \(执行角色\)](#)。

2. 打开文本编辑器，并用以下内容创建名为 helloWorld.js 的文件：

```
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. 压缩 helloWorld.js 文件，并用名称 helloWorld.zip 保存它。
4. 使用以下命令，其中的角色是您在第一步中设置的 Lambda 执行角色：

```

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. 授予 CloudWatch Logs 执行您的函数的权限。使用以下命令，将占位符账户替换为您自己的账户。

```

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. 使用以下命令创建账户级订阅筛选策略，将占位符账户替换为您自己的账户。在此示例中，所有包含字符串 ERROR 的日志事件都将流式传输，但名为 LogGroupToExclude1 和 LogGroupToExclude2 的日志组中的日志事件除外。

```

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"
```

设置订阅筛选条件后，CloudWatch Logs 会将与筛选条件模式和选择标准匹配的所有传入事件日志转发到您的流。

selection-criteria 字段是可选的，但对于从订阅筛选条件中排除可能导致无限日志递归的日志组非常重要。有关此问题以及确定要排除哪些日志组的更多信息，请参阅 [防止日志递归](#)。目前，NOT IN 是 selection-criteria 唯一支持的运算符。

7. (可选) 使用样本日志事件进行测试。在出现命令提示符时，运行以下命令，以将一条简单的日志消息放入已订阅的流中。

要查看 Lambda 函数的输出，请导航至 Lambda 函数，在此处可以查看 /aws/lambda/helloworld 中的输出：

```
aws logs put-log-events --log-group-name Example1 --log-stream-name logStream1 --  
log-events "[{\"timestamp\":CURRENT_TIMESTAMP_MILLIS , \"message\": \"Simple Lambda  
Test\"}]"
```

您将看到包含一组 Lambda 的响应。Lambda 记录中的数据属性采用 base64 编码并使用 gzip 格式进行压缩。Lambda 接收的实际负载采用以下格式：{ "awslogs": { "data": "BASE64ENCODED_GZIP_COMPRESSED_DATA" } }。您可以使用以下 Unix 命令检查命令行中的原始数据：

```
echo -n "<BASE64ENCODED_GZIP_COMPRESSED_DATA>" | base64 -d | zcat
```

Base64 解码和解压缩数据被格式化为 JSON 并具有以下结构：

```
{  
  "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

账户级订阅筛选条件将不会应用于目标 Lambda 函数的日志组。这是为了防止可能导致提取计费增加的无限日志递归。有关此问题的更多信息，请参阅 [防止日志递归](#)。

数据架构中的关键元素如下：

messageType

数据消息将使用“DATA_MESSAGE”类型。有时候，CloudWatch Logs 可能发出类型为“CONTROL_MESSAGE”的 Kinesis Data Streams 记录，这主要是为了检查目标是否可访问。

owner

原始生成日志数据的 Amazon 账户 ID。

logGroup

原始日志数据的日志组名称。

logStream

原始日志数据的日志流名称。

subscriptionFilters

与原始日志数据匹配的订阅筛选条件名称的列表。

logEvents

表示为一组日志事件记录的实际日志数据。“id”属性是每个日志事件的唯一标识符。

policyLevel

强制实施策略的级别。“ACCOUNT_LEVEL_POLICY”是账户级订阅筛选策略的 policyLevel。

示例 3：Amazon Data Firehose 订阅筛选条件

此示例将创建一个 CloudWatch Logs 账户级订阅筛选策略，用于将与所定义筛选条件匹配的传入日志事件发送到您的 Amazon Data Firehose 传输流。已经使用 gzip 6 级压缩对从 CloudWatch Logs 发送到 Amazon Data Firehose 的数据进行压缩，因此无需在 Firehose 传输流中使用压缩。然后，可以使用 Firehose 中的解压缩功能自动解压缩日志。有关更多信息，请参阅 [Writing to Kinesis Data Firehose Using CloudWatch Logs](#)。

Warning

在创建 Firehose 流之前，请计算将生成的日志数据量。请确保创建一个处理此数据量的 Firehose 流。如果流无法处理卷，则日志流将受限制。有关 Firehose 流数据量限制的更多信息，请参阅 [Amazon Data Firehose Data Limits](#)。

为 Firehose 创建订阅筛选条件

1. 创建 Amazon Simple Storage Service (Amazon S3) 存储桶 我们建议您使用专为 CloudWatch Logs 创建的存储桶。但是，如果要使用现有存储桶，请跳至第 2 步。

运行以下命令，将占位符区域替换为您想使用的区域：

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket2 --create-bucket-configuration  
LocationConstraint=region
```

下面是示例输出：

```
{
  "Location": "/amzn-s3-demo-bucket2"
}
```

2. 创建 IAM 角色，该角色将授予 Amazon Data Firehose 将数据放入您的 Amazon S3 存储桶的权限。

有关更多信息，请参阅《Amazon Data Firehose 开发人员指南》中的 [Controlling Access with Amazon Data Firehose](#)。

首先，使用文本编辑器在文件 `~/TrustPolicyForFirehose.json` 中创建一个信任策略，具体如下所示：

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "firehose.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

3. 使用 `create-role` 命令创建 IAM 角色，并指定信任策略文件。记下返回的 `Role.Arn` 值，因为后面的步骤中将会用到它：

```
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. 创建权限策略以定义 Firehose 可对您的账户执行的操作。首先，使用文本编辑器在文件 `~/PermissionsForFirehose.json` 中创建权限策略：

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:PutObject" ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket2",
        "arn:aws:s3:::amzn-s3-demo-bucket2/*" ]
    }
  ]
}

```

5. 使用以下 `put-role-policy` 命令将权限策略与角色关联：

```
aws iam put-role-policy --role-name FirehoseToS3Role --policy-name Permissions-
Policy-For-Firehose --policy-document file://~/PermissionsForFirehose.json
```

6. 如下所示，创建目标 Firehose 传输流，将 RoleARN 和 BucketARN 的占位符值替换为您创建的角色和存储桶 ARN：

```
aws firehose create-delivery-stream \
  --delivery-stream-name 'my-delivery-stream' \
  --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::123456789012:role/FirehoseToS3Role", "BucketARN":
  "arn:aws:s3:::amzn-s3-demo-bucket2"}'
```

NFirehose 对所传输的 Amazon S3 对象自动使用 YYYY/MM/DD/HH UTC 时间格式的前缀。您可以指定要添加在该时间格式前缀前面的额外前缀。如果前缀以正斜杠 (/) 结束，则在 Amazon S3 存储桶中显示为文件夹。

7. 等待几分钟让流变为活动状态。您可以使用 Firehose `describe-delivery-stream` 命令检查 `DeliveryStreamDescription.DeliveryStreamStatus` 属性。此外，请记住 `DeliveryStreamDescription.DeliveryStreamARN` 值，因为后面的步骤中将会用到它：

```
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-0000000000001",
        "S3DestinationDescription": {
          "CompressionFormat": "UNCOMPRESSED",
          "EncryptionConfiguration": {
            "NoEncryptionConfig": "NoEncryption"
          },
          "RoleARN": "delivery-stream-role",
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket2",
          "BufferingHints": {
            "IntervalInSeconds": 300,
            "SizeInMBs": 5
          }
        }
      }
    ]
  }
}
```

8. 创建 IAM 角色，该角色将授予 CloudWatch Logs 将数据放入 Firehose 传输流的权限。首先，使用文本编辑器在文件 `~/TrustPolicyForCWL.json` 中创建信任策略：

此策略包括 `aws:SourceArn` 全局条件上下文密钥，有助于避免出现混淆代理安全问题。有关更多信息，请参阅 [混淆代理问题防范](#)。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringLike": {
        "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
      }
    }
  }
}
```

9. 使用 `create-role` 命令创建 IAM 角色，并指定信任策略文件。记下返回的 `Role.Arn` 值，因为后面的步骤中将会用到它：

```
aws iam create-role \
--role-name CWLtoKinesisFirehoseRole \
--assume-role-policy-document file:///~/TrustPolicyForCWL.json

{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
          "Service": "logs.amazonaws.com"
        },
        "Condition": {
          "StringLike": {
            "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
          }
        }
      }
    },
    "RoleId": "AA0IIAH450GAB4HC5F431",
    "CreateDate": "2015-05-29T13:46:29.431Z",
    "RoleName": "CWLtoKinesisFirehoseRole",
```

```

    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
  }
}

```

10. 创建权限策略以定义可对您的账户执行的 CloudWatch Logs 操作。首先，使用文本编辑器创建权限策略文件（例如 `~/PermissionsForCWL.json`）：

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["firehose:PutRecord"],
      "Resource": [
        "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-
name"]
      }
    ]
  }
}

```

11. 使用 `put-role-policy` 命令将权限策略与角色关联：

```

aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-
name Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json

```

12. 在 Amazon Data Firehose 传输流进入活动状态并且您已创建 IAM 角色后，便可以创建 CloudWatch Logs 账户级订阅筛选策略。该策略将立即启动从所选日志组到您的 Amazon Data Firehose 传输流的实时日志数据流动：

```

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. 当您设置订阅筛选条件后，CloudWatch Logs 会将与筛选条件模式匹配的传入事件日志转发到您的 Amazon Data Firehose 传输流。

`selection-criteria` 字段是可选的，但对于从订阅筛选条件中排除可能导致无限日志递归的日志组非常重要。有关此问题以及确定要排除哪些日志组的更多信息，请参阅 [防止日志递归](#)。目前，`NOT IN` 是 `selection-criteria` 唯一支持的运算符。

根据 Amazon Data Firehose 传输流上设置的时间缓冲间隔，数据将开始显示在 Amazon S3 中。经过足够的时间后，您可以通过检查您的 Amazon S3 存储桶验证您的数据。

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket2' --prefix 'firehose/'
{
  "Contents": [
    {
      "LastModified": "2023-10-29T00:01:25.000Z",
      "ETag": "\"a14589f8897f4089d3264d9e2d1f1610\"",
      "StorageClass": "STANDARD",
      "Key": "firehose/2015/10/29/00/my-delivery-stream-2015-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250",
      "Owner": {
        "DisplayName": "cloudwatch-logs",
        "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b5"
      },
      "Size": 593
    },
    {
      "LastModified": "2015-10-29T00:35:41.000Z",
      "ETag": "\"a7035b65872bb2161388ffb63dd1aec5\"",
      "StorageClass": "STANDARD",
      "Key": "firehose/2023/10/29/00/my-delivery-stream-2023-10-29-00-35-40-EXAMPLE-7e66-49bc-9fd4-fc9819cc8ed3",
      "Owner": {
        "DisplayName": "cloudwatch-logs",
        "ID": "EXAMPLE6be062b19584e0b7d84ecc19237f87b6"
      },
      "Size": 5752
    }
  ]
}
```

```
aws s3api get-object --bucket 'amzn-s3-demo-bucket2' --key 'firehose/2023/10/29/00/my-delivery-stream-2023-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250'
testfile.gz
```

```
{
  "AcceptRanges": "bytes",
  "ContentType": "application/octet-stream",
  "LastModified": "Thu, 29 Oct 2023 00:07:06 GMT",
  "ContentLength": 593,
  "Metadata": {}
}
```

Amazon S3 对象中的数据以 gzip 格式压缩。您可以使用以下 Unix 命令检查命令行中的原始数据：

```
zcat testfile.gz
```

跨账户跨区域订阅

您可以与不同 Amazon 账户的所有者合作，并在您的 Amazon 资源（例如 Amazon Kinesis 或 Amazon Data Firehose 流）上接收他们的日志事件（这称为“跨账户数据共享”）。例如，此日志事件数据可从集中的 Kinesis Data Streams 或 Firehose 流中读取以执行自定义处理和分析。自定义处理在您跨多个账户协作和分析数据时特别有用。

例如，某家公司的信息安全组可能需要分析数据以进行实时入侵检测或查找异常行为，以便能对公司所有部门的账户进行审核（通过收集各账户的联合生产日志进行集中处理）。可以汇编这些账户中的事件数据的实时流并将其传递到信息安全组，这些组可使用 Kinesis Data Streams 将数据附加到现有安全分析系统。

Note

日志组和目标必须位于同一 Amazon 区域内。但是，目标指向的 Amazon 资源可位于不同的区域。在以下部分的示例中，所有特定于区域的资源都是在美国东部（弗吉尼亚州北部）创建的。

如果您已配置 Amazon Organizations 并且正在使用成员账户，则可以使用日志集中化将源账户中的日志数据收集到中央监控账户中。

如果使用集中式日志组，则当创建订阅筛选条件时，您可以使用以下系统字段维度：

- `@aws.account` - 此维度表示日志事件来源的 Amazon 账户 ID。

- `@aws.region` - 此维度表示生成日志事件的 Amazon 区域。

这些维度有助于识别日志数据的来源，从而对从集中式日志派生的指标进行更精细的筛选和分析。

主题

- [使用 Kinesis Data Streams 进行跨账户跨区域日志数据共享](#)
- [使用 Firehose 进行跨账户跨区域日志数据共享](#)
- [使用 Kinesis Data Streams 进行跨账户跨区域账户级订阅](#)
- [使用 Firehose 进行跨账户跨区域账户级订阅](#)

使用 Kinesis Data Streams 进行跨账户跨区域日志数据共享

创建跨账户订阅时，可以指定单个账户或企业作为发件人。如果指定企业，则此过程允许企业中的所有账户向接收方账户发送日志。

要跨账户共享日志数据，您需要建立日志数据发送者和接收者：

- Log data sender (日志数据发件人) – 从收件人处获取目标信息并告知 CloudWatch Logs 发件人已准备好将其录入事件发送到指定目标。在本部分其余程序中，日志数据发送者显示为虚构的 Amazon 账号 111111111111。

如果您将在企业中设立多个账户向收件人账户发送日志，则可以创建策略，授予企业中的所有账户向收件人账户发送日志的权限。您仍然必须为每个发件人账户设置单独的订阅筛选条件。

- 日志数据接收者 – 设置一个封装 Kinesis Data Streams 流的目标，并告知 CloudWatch Logs 接收者希望接收日志数据。接收者随后与发送者共享与此目标有关的信息。在本部分其余程序中，日志数据接收者显示为虚构的 Amazon 账号 999999999999。

要开始接收来自跨账户用户的日志事件，日志数据接收者应首先创建一个 CloudWatch Logs 目标。每个目标都包含以下键元素：

目标名称

您要创建的目标的名称。

目标 ARN

您要用作订阅源目标的 Amazon 资源的 Amazon Resource Name (ARN)。

角色 ARN

一个 Amazon Identity and Access Management (IAM) 角色，用于向 CloudWatch Logs 授予将数据放入所选流所需的必要权限。

访问策略

一个 IAM policy 文档（采用 JSON 格式，使用 IAM policy 语法编写），用于管理有权对您的目标进行写入的用户组。

Note

日志组和目标必须位于同一 Amazon 区域内。但是，目标指向的 Amazon 资源可位于不同的区域。在以下部分的示例中，所有特定于区域的资源都是在美国东部（弗吉尼亚北部）创建的。

主题

- [设置新的跨账户订阅](#)
- [更新现有的跨账户订阅](#)

设置新的跨账户订阅

按照这些部分中所述步骤设置新的跨账户日志订阅。

主题

- [步骤 1：创建目标](#)
- [步骤 2：（仅在使用企业时）创建 IAM 角色](#)
- [步骤 3：添加/验证跨账户目标的 IAM 权限](#)
- [步骤 4：创建订阅筛选条件](#)
- [验证日志事件流](#)
- [在运行时修改目标成员资格](#)

步骤 1：创建目标

Important

此过程中的所有步骤都需要在日志数据接收者账户中完成。

在本示例中，日志数据接收者账户的 Amazon 账户 ID 为 999999999999，而日志数据发送者 Amazon 账户 ID 为 111111111111。

该示例使用名为 RecipientStream 的 Kinesis Data Streams 流创建一个目标，并创建一个允许 CloudWatch Logs 将数据写入到该目标的角色。

创建目标时，CloudWatch Logs 将代表收件人账户向目标发送测试消息。当订阅筛选条件稍后处于活动状态时，CloudWatch Logs 会代表源账户向目标发送录入事件。

创建目标

1. 在收件人账户中，在 Kinesis Data Streams 中创建目标流。在命令提示符下，输入：

```
aws kinesis create-stream --stream-name "RecipientStream" --shard-count 1
```

2. 等到 流变为活动状态。您可使用 `aws kinesis describe-stream` 命令检查 `StreamDescription.StreamStatus` 属性。此外，请记下 `StreamDescription.StreamARN` 的值，因为稍后您需要将该值传递到 CloudWatch Logs：

```
aws kinesis describe-stream --stream-name "RecipientStream"
{
  "StreamDescription": {
    "StreamStatus": "ACTIVE",
    "StreamName": "RecipientStream",
    "StreamARN": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "34028236692093846346337460743176EXAMPLE",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
            "4955113521868881845667950383198145878459135270218EXAMPLE"
        }
      }
    ]
  }
}
```

```

    }
  }
]
}
}

```

您的流可能需要一两分钟才会以活动状态显示。

3. 创建 IAM 角色，该角色将授予 CloudWatch Logs 将数据放入流的权限。首先，您需要在文件 `~/TrustPolicyForCWL.json` 中创建信任策略。使用文本编辑器创建此策略文件，请勿使用 IAM 控制台来创建。

此策略包括指定 `sourceAccountId` 的 `aws:SourceArn` 全局条件上下文密钥，有助于避免出现混淆代理安全问题。如果您在第一次调用中还不知道源账户 ID，则建议您将目标 ARN 放在源 ARN 字段中。在随后的调用中，应将源 ARN 设置为从第一次调用中收集的实际源 ARN。有关更多信息，请参阅 [混淆代理问题防范](#)。

```

{
  "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. 使用 `aws iam create-role` 命令创建 IAM 角色，并指定信任策略文件。记下返回的 `Role.Arn` 值，因为该值稍后也将传递到 CloudWatch Logs：

```

aws iam create-role \
--role-name CWLtoKinesisRole \
--assume-role-policy-document file://~/TrustPolicyForCWL.json

```

```
{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Condition": {
          "StringLike": {
            "aws:SourceArn": [
              "arn:aws:logs:region:sourceAccountId:*",
              "arn:aws:logs:region:recipientAccountId:"
            ]
          }
        },
        "Principal": {
          "Service": "logs.amazonaws.com"
        }
      }
    },
    "RoleId": "AA0IIAH450GAB4HC5F431",
    "CreateDate": "2015-05-29T13:46:29.431Z",
    "RoleName": "CWLtoKinesisRole",
    "Path": "/",
    "Arn": "arn:aws:iam::999999999999:role/CWLtoKinesisRole"
  }
}
```

5. 创建权限策略以定义 CloudWatch Logs 可对您的账户执行的操作。首先，使用文本编辑器在文件 `~/PermissionsForCWL.json` 中创建权限策略：

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:999999999999:stream/RecipientStream"
    }
  ]
}
```

6. 使用 `aws iam put-role-policy` 命令将权限策略与角色关联：

```
aws iam put-role-policy \
```

```
--role-name CWLtoKinesisRole \
--policy-name Permissions-Policy-For-CWL \
--policy-document file://~/PermissionsForCWL.json
```

7. 在流变为活动状态并且您已创建 IAM 角色后，您便可以创建 CloudWatch Logs 目标。
- a. 此步骤不会将访问策略与您的目标关联，它只是完成目标创建的两个步骤中的第一个步骤。记下负载中返回的 DestinationArn：

```
aws logs put-destination \
  --destination-name "testDestination" \
  --target-arn "arn:aws:kinesis:region:999999999999:stream/RecipientStream" \
  --role-arn "arn:aws:iam::999999999999:role/CWLtoKinesisRole"

{
  "DestinationName" : "testDestination",
  "RoleArn" : "arn:aws:iam::999999999999:role/CWLtoKinesisRole",
  "DestinationArn" : "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
  "TargetArn" : "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream"
}
```

- b. 在步骤 7a 完成后，在日志数据接收者账户中将访问策略与目标关联。此策略必须指定 logs:PutSubscriptionFilter 操作并向发件人账户授予权限才能访问目标。

此策略向发送日志的 Amazon 账户授予权限。您可以在策略中仅指定这一个账户，如果发件人账户是企业的成员，则策略可以指定企业的企业 ID。这样，您只能创建策略，以允许企业中的多个账户向此目标账户发送日志。

使用文本编辑器创建名为 ~/AccessPolicy.json 的文件，并包含下列策略语句之一。

第一个示例策略允许企业中所有 ID 为 o-1234567890 的账户将日志发送到收件人账户。

JSON

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

```

        "Action": "logs:PutSubscriptionFilter",
        "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
        "Condition": {
            "StringEquals": {
                "aws:PrincipalOrgID": [
                    "o-1234567890"
                ]
            }
        }
    }
}

```

下一个示例只允许日志数据发件人账户 (111111111111) 将日志发送到日志数据收件人账户。

JSON

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111111111111"
            },
            "Action": "logs:PutSubscriptionFilter",
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination"
        }
    ]
}

```

- c. 将您在上一步中创建的策略附加到目标。

```

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

```

此访问策略允许 ID 为 111111111111 的 Amazon 账户中的用户针对 ARN 为 `arn:aws:logs:region:999999999999:destination:testDestination` 的目标调用 `PutSubscriptionFilter`。任何其他用户针对此目标调用 `PutSubscriptionFilter` 的尝试都会被拒绝。

要针对访问策略验证用户的权限，请参阅 IAM 用户指南中的[使用策略验证程序](#)。

完成后，如果您为跨账户权限使用的是 Amazon Organizations，请按照[步骤 2：（仅在使用企业时）创建 IAM 角色](#)中的步骤操作。如果您选择直接向另一个账户授予权限而不是使用 Organizations，则可以跳过该步骤然后继续执行[步骤 4：创建订阅筛选条件](#)。

步骤 2：（仅在使用企业时）创建 IAM 角色

在上一节中，如果您使用向账户 111111111111 所在的企业授予权限的访问策略创建了目标，而不是直接向账户 111111111111 授予权限，则请按照本节中的步骤操作。否则，您可以跳至[步骤 4：创建订阅筛选条件](#)。

本节中的步骤创建了 IAM 角色，CloudWatch 可以代入该角色，并验证发件人账户是否有权针对收件人目标创建订阅筛选条件。

在发送者账户中执行本节中的步骤。该角色必须存在于发送者账户中，并且您在订阅筛选器中指定该角色的 ARN。在此示例中，发送者账户为 111111111111。

要使用 Amazon Organizations 为跨账户日志订阅创建所需的 IAM 角色

1. 请在文件 `/TrustPolicyForCWLSubscriptionFilter.json` 中创建以下信任策略。使用文本编辑器创建此策略文件；请勿使用 IAM 控制台来创建。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

2. 创建一个使用此策略的 IAM 角色。记录该命令返回的 `Arn` 值，您需要在本过程的后续部分中使用该值。在此示例中，我们将 `CWLtoSubscriptionFilterRole` 用作我们所创建角色的名称。

```
aws iam create-role \
```



```
--role-name CWLtoSubscriptionFilterRole \  
--assume-role-policy-document file://~/  
TrustPolicyForCWLSubscriptionFilter.json
```

3. 创建权限策略以定义 CloudWatch Logs 可对您的账户执行的操作。

- a. 首先，使用文本编辑器在名为 `~/PermissionsForCWLSubscriptionFilter.json` 的文件中创建以下权限策略。

```
{  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "logs:PutLogEvents",  
      "Resource": "arn:aws:logs:region:111111111111:log-  
group:LogGroupOnWhichSubscriptionFilterIsCreated:*"  
    }  
  ]  
}
```

- b. 输入以下命令，以将刚创建的权限策略与您在步骤 2 中创建的角色相关联。

```
aws iam put-role-policy  
--role-name CWLtoSubscriptionFilterRole  
--policy-name Permissions-Policy-For-CWL-Subscription-filter  
--policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

完成后，您可以继续执行 [步骤 4：创建订阅筛选条件](#)。

步骤 3：添加/验证跨账户目标的 IAM 权限

根据 Amazon 跨账户策略评估逻辑，为访问任何跨账户资源（例如用作订阅筛选条件目标的 Kinesis 或 Firehose 流），您必须在发送账户中设置基于身份的策略，该策略提供对跨账户目标资源的明确访问权限。有关策略评估逻辑的更多信息，请参阅[跨账户策略评估逻辑](#)。

您可以将基于身份的策略附加到用于创建订阅筛选条件的 IAM 角色或 IAM 用户。此策略必须位于发送账户中。如果您使用管理员角色创建订阅筛选条件，则可以跳过此步骤继续前进至 [步骤 4：创建订阅筛选条件](#)。

添加或验证跨账户所需的 IAM 权限

1. 输入以下命令以检查使用哪个 IAM 角色或 IAM 用户来运行 Amazon 日志命令。

```
aws sts get-caller-identity
```

该命令返回的输出类似于下方内容：

```
{
  "UserId": "User ID",
  "Account": "sending account id",
  "Arn": "arn:aws:sending account id:role/user:RoleName/UserName"
}
```

记下由 *RoleName* 或 *UserName* 表示的值。

2. 登录发送账户中的 Amazon Web Services 管理控制台，使用您在步骤 1 所输入命令的输出中返回的 IAM 角色或 IAM 用户来搜索附加的策略。
3. 验证附加到该角色或用户的策略是否提供在跨账户目标资源上调用 `logs:PutSubscriptionFilter` 的明确权限。

以下策略提供权限，仅在单个 Amazon 账户（账户 123456789012）中对任何目标资源创建订阅筛选条件：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSubscriptionFiltersOnAccountResources",
      "Effect": "Allow",
      "Action": "logs:PutSubscriptionFilter",
      "Resource": [
        "arn:aws:logs::*:log-group:*",
        "arn:aws:logs::123456789012:destination:*"
      ]
    }
  ]
}
```

以下策略提供权限，仅在单个 Amazon 账户（账户 123456789012）中对名为 sampleDestination 的特定目标资源创建订阅筛选条件：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSubscriptionFilteronAccountResource",
      "Effect": "Allow",
      "Action": "logs:PutSubscriptionFilter",
      "Resource": [
        "arn:aws:logs::*:log-group:*",
        "arn:aws:logs:*:123456789012:destination:sampleDestination"
      ]
    }
  ]
}
```

步骤 4：创建订阅筛选条件

在创建目标后，日志数据接收者账户可与其他 Amazon 账户共享目标 ARN (arn:aws:logs:us-east-1:999999999999:destination:testDestination)，以便他们将日志事件发送到相同的目标。这些其他的发送账户用户随后会在各自的日志组上针对此目标创建订阅筛选条件。订阅筛选器将立即让实时日志数据开始从所选日志组向指定目标的流动。

Note

如果要向整个组织授予订阅筛选器的权限，则需要使用在 [步骤 2：（仅在使用企业时）创建 IAM 角色](#) 中创建的 IAM 角色的 ARN。

在以下示例中，在发送者账户中创建订阅筛选条件。此筛选条件与一个包含 Amazon CloudTrail 事件的日志组关联，以便将“根”Amazon 凭证执行的每个记录下来的活动都传输到前面创建的目标。该目标封装一个名为“RecipientStream”的流。

以下各部分中的其余步骤假定您已按照《Amazon CloudTrail 用户指南》中[将 CloudTrail 事件发送到 CloudWatch Logs](#) 中指示操作，并已创建包含 CloudTrail 事件的日志组。这些步骤假定此日志组的名称为 CloudTrail/logs。

输入以下命令时，请务必以 IAM 用户身份登录，或者使用在 [步骤 3：添加/验证跨账户目标的 IAM 权限](#) 中为其添加策略的 IAM 角色登录。

```
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"
```

日志组和目标必须位于同一 Amazon 区域内。但是，目标可指向位于不同区域的 Amazon 资源，如 Kinesis Data Streams 流。

验证日志事件流

在创建订阅筛选条件后，CloudWatch Logs 会将与筛选条件模式匹配的所有传入日志事件转发至封装在名为“RecipientStream”的目标流中的流。目标所有者可以使用 `aws kinesis get-shard-iterator` 命令抓取 Kinesis Data Streams 分片，并使用 `aws kinesis get-records` 命令提取一些 Kinesis Data Streams 记录以验证是否发生该过程：

```
aws kinesis get-shard-iterator \  
  --stream-name RecipientStream \  
  --shard-id shardId-000000000000 \  
  --shard-iterator-type TRIM_HORIZON  
  
{  
  "ShardIterator":  
    "AAAAAAAAAAFGU/  
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev  
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f  
+0IK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"  
}  
  
aws kinesis get-records \  
  --limit 10 \  
  --shard-iterator  
    "AAAAAAAAAAFGU/  
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
```

```
+e2P4djJg4L9wmXKvQYoE+rMUIFq+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+0IK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
```

Note

您可能需要在 Kinesis Data Streams 开始返回数据前几分钟重新运行 `get-records` 命令。

您将看到包含一组 Kinesis Data Streams 记录的响应。Kinesis Data Streams 记录中的数据属性使用 gzip 格式压缩并采用 Base64 编码。您可以使用以下 Unix 命令检查命令行中的原始数据：

```
echo -n "<Content of Data>" | base64 -d | zcat
```

Base64 解码和解压缩数据被格式化为 JSON 并具有以下结构：

```
{
  "owner": "111111111111",
  "logGroup": "CloudTrail/logs",
  "logStream": "111111111111_CloudTrail/logs_us-east-1",
  "subscriptionFilters": [
    "RecipientStream"
  ],
  "messageType": "DATA_MESSAGE",
  "logEvents": [
    {
      "id": "3195310660696698337880902507980421114328961542429EXAMPLE",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
    }",
    },
    {
      "id": "3195310660696698337880902507980421114328961542429EXAMPLE",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
    }",
    },
    {
      "id": "3195310660696698337880902507980421114328961542429EXAMPLE",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
    }"
  ]
}
```

```
]
}
```

此数据结构中的键元素如下所示：

owner

原始生成日志数据的 Amazon 账户 ID。

logGroup

原始日志数据的日志组名称。

logStream

原始日志数据的日志流名称。

subscriptionFilters

与原始日志数据匹配的订阅筛选条件名称的列表。

messageType

数据消息将使用“DATA_MESSAGE”类型。有时候，CloudWatch Logs 可能发出类型为“CONTROL_MESSAGE”的 Kinesis Data Streams 记录，这主要是为了检查目标是否可访问。

logEvents

表示为一组日志事件记录的实际日志数据。ID 属性是每个日志事件的唯一标识符。

在运行时修改目标成员资格

您可能遇到必须在您拥有的目标中添加或删除某些用户的成员资格的情况。您可通过新访问策略对您的目标使用 `put-destination-policy` 命令。在以下示例中，将阻止之前添加的账户 111111111111 再发送任何日志数据，并将启用账户 222222222222。

1. 提取当前与目标 `testDestination` 关联的策略并记下 `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. 更新该策略，以体现已阻止账户 111111111111，并且账户 222222222222 已启用。将此策略放入 ~/NewAccessPolicy.json 文件：

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "222222222222"
      },
      "Action": "logs:PutSubscriptionFilter",
      "Resource": "arn:aws:logs:us-
      east-1:999999999999:destination:testDestination"
    }
  ]
}

```

3. 调用 PutDestinationPolicy 以将 NewAccessPolicy.json 文件中定义的策略与目标关联：

```

aws logs put-destination-policy \
--destination-name "testDestination" \
--access-policy file://~/NewAccessPolicy.json

```

这将最终禁用账户 ID 111111111111 中的日志事件。在账户 222222222222 的所有者创建订阅筛选条件后，账户 ID 222222222222 中的日志事件就会立即开始流向目标。

更新现有的跨账户订阅

如果您当前有跨账户日志订阅，其中目标账户仅向特定发件人账户授予权限，并且您想更新此订阅以便目标账户向企业中所有账户授予访问权限，请按照本节中的步骤操作。

主题

- [步骤 1：更新订阅筛选条件](#)
- [步骤 2：更新现有的目标访问策略](#)

步骤 1：更新订阅筛选条件

Note

此步骤仅对于由 [启用从 Amazon 服务进行日志记录](#) 中列出的服务创建的日志的跨账户订阅才需要。如果您不使用由这些日志组之一创建的日志，则可以跳至 [步骤 2：更新现有的目标访问策略](#)。

在某些情况下，您必须更新向目标账户发送日志的所有发件人账户中的订阅筛选条件。此更新添加了 IAM 角色，CloudWatch 可以承担该角色，并验证发件人账户是否有向收件人账户发送日志的权限。

请按照本节中的步骤为您要更新的每个发件人账户执行操作，以使用企业 ID 获得跨账户订阅权限。

在本节的示例中，111111111111 和 222222222222 两个账户已经创建了用于向账户 999999999999 发送日志的订阅筛选条件。现有的订阅筛选条件值如下：

```
## 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"
```

如果需要查找当前订阅筛选条件参数值，请输入以下命令。

```
aws logs describe-subscription-filters
\ --log-group-name "my-log-group-name"
```


要更新订阅筛选条件以开始使用企业 ID 获取跨账户日志权限

1. 请在文件 `~/TrustPolicyForCWL.json` 中创建以下信任策略。使用文本编辑器创建此策略文件；请勿使用 IAM 控制台来创建。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

2. 创建一个使用此策略的 IAM 角色。记录该命令返回的 Arn 值的 Arn 值，您将需要在本过程的后面部分中使用该值。在此示例中，我们将 `CWLtoSubscriptionFilterRole` 用作我们所创建角色的名称。

```
aws iam create-role
\ --role-name CWLtoSubscriptionFilterRole
\ --assume-role-policy-document file://~/TrustPolicyForCWL.json
```

3. 创建权限策略以定义 CloudWatch Logs 可对您的账户执行的操作。
 - a. 首先，使用文本编辑器在名为 `/PermissionsForCWLSubscriptionFilter.json` 的文件中创建以下权限策略。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:PutLogEvents",
      "Resource": "arn:aws:logs:region:111111111111:log-group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
    }
  ]
}
```

- b. 输入以下命令，以将刚创建的权限策略与您在步骤 2 中创建的角色相关联。

```
aws iam put-role-policy
--role-name CWLtoSubscriptionFilterRole
--policy-name Permissions-Policy-For-CWL-Subscription-filter
```

```
--policy-document file:///~/PermissionsForCWLSubscriptionFilter.json
```

4. 输入以下命令以更新订阅筛选条件。

```
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"
```

步骤 2：更新现有的目标访问策略

更新所有发件人账户中的订阅筛选条件后，可以更新收件人账户中的目标访问策略。

在以下示例中，收件人账户为 999999999999，目的地名为 testDestination。

此更新将使企业中 ID 为 o-1234567890 的所有账户向收件人账户发送日志。只有创建了订阅筛选条件的账户才会实际向收件人账户发送日志。

要更新收件人账户中的目标访问策略以开始使用企业 ID 获取权限

1. 在收件人账户中，请使用文本编辑器创建包含下列内容的 ~/AccessPolicy.json 文件。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "logs:PutSubscriptionFilter",
      "Resource": "arn:aws:logs:us-east-1:999999999999:destination:testDestination",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": [
            "o-1234567890"
          ]
        }
      }
    }
  ]
}
```

```
    ]
  }
}
}
```

2. 输入以下命令，将刚创建的策略附加到现有目标。要更新目标以使用具有企业 ID 的访问策略而不是列出特定 Amazon 账户 ID 的访问策略，请包括 `force` 参数。

Warning

如果您正在使用由 Amazon 中列出的 [启用从 Amazon 服务进行日志记录](#) 服务发送的日志，则在执行此步骤之前，您必须先按照 [步骤 1：更新订阅筛选条件](#) 中的说明更新所有发件人账户中的订阅筛选条件。

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

使用 Firehose 进行跨账户跨区域日志数据共享

要跨账户共享日志数据，您需要建立日志数据发送者和接收者：

- 日志数据发送者 – 从接收者处获取目标信息并告知 CloudWatch Logs 发送者已准备好将其日志事件发送到指定目标。在本部分其余程序中，日志数据发送者显示为虚构的 Amazon 账号 111111111111。
- 日志数据接收者 – 设置一个封装 Kinesis Data Streams 流的目标，并告知 CloudWatch Logs 接收者希望接收日志数据。接收者随后与发送者共享与此目标有关的信息。在本部分其余程序中，日志数据接收者显示为虚构的 Amazon 账号 222222222222。

本节中的示例使用带有 Amazon S3 存储的 Firehose 传输流。您还可以使用不同的设置来设置 Firehose 传输流。有关更多信息，请参阅 [Creating a Firehose Delivery Stream](#)。

 Note

日志组和目标必须位于同一Amazon区域内。但是，目标指向的Amazon资源可位于不同的区域。

 Note

支持同一账户和跨区域传输流的 Firehose 订阅筛选条件。

主题

- [步骤 1：创建 Firehose 传输流](#)
- [步骤 2：创建目标](#)
- [步骤 3：添加/验证跨账户目标的 IAM 权限](#)
- [步骤 4：创建订阅筛选条件](#)
- [验证录入事件流](#)
- [在运行时修改目标成员资格](#)

步骤 1：创建 Firehose 传输流

 Important

在完成以下步骤之前，您必须使用访问策略，以便 Firehose 可以访问您的 Amazon S3 存储桶。有关更多信息，请参阅《Amazon Data Firehose 开发人员指南》中的 [Controlling Access](#)。

本部分（步骤 1）中的所有步骤都需要在日志数据接收者账户中完成。

以下示例命令中使用的是美国东部（弗吉尼亚州北部）。请将此区域替换为适用于您的部署的正确区域。

创建将用作目标的 Firehose 传输流

1. 创建 Amazon S3 存储桶：

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --create-bucket-configuration
LocationConstraint=us-east-1
```

2. 创建 IAM 角色，授予 Firehose 将数据放入存储桶的权限。

- a. 首先，使用文本编辑器在文件 `~/TrustPolicyForFirehose.json` 中创建信任策略。

```
{ "Statement": { "Effect": "Allow", "Principal": { "Service":
  "firehose.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition":
  { "StringEquals": { "sts:ExternalId": "222222222222" } } } }
```

- b. 创建 IAM 角色，并指定您刚创建的信任策略文件。

```
aws iam create-role \
  --role-name FirehoseToS3Role \
  --assume-role-policy-document file://~/TrustPolicyForFirehose.json
```

- c. 此命令的输出与以下内容类似。记下角色名称和角色 ARN。

```
{
  "Role": {
    "Path": "/",
    "RoleName": "FirehoseToS3Role",
    "RoleId": "AR0AR3BXASEKW7K635M53",
    "Arn": "arn:aws:iam::222222222222:role/FirehoseToS3Role",
    "CreateDate": "2021-02-02T07:53:10+00:00",
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Effect": "Allow",
        "Principal": {
          "Service": "firehose.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
          "StringEquals": {
            "sts:ExternalId": "222222222222"
          }
        }
      }
    }
  }
}
```

```
}
```

3. 创建权限策略，以定义 Firehose 可对您的账户执行的操作。

- a. 首先，使用文本编辑器在名为 `~/PermissionsForFirehose.json` 的文件中创建以下权限策略。根据应用场景，您可能需要为此文件添加更多权限。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:ListBucket"
    ],
    "Resource": [
      "arn:aws:s3:::amzn-s3-demo-bucket",
      "arn:aws:s3:::amzn-s3-demo-bucket/*"
    ]
  }]
}
```

- b. 输入以下命令，以将刚创建的权限策略与 IAM 角色相关联。

```
aws iam put-role-policy --role-name FirehoseToS3Role --policy-name
Permissions-Policy-For-Firehose-To-S3 --policy-document file://~/
PermissionsForFirehose.json
```

4. 输入以下命令以创建 Firehose 传输流。将 *my-role-arn* 和 *amzn-s3-demo-bucket2-arn* 替换为正确的部署值。

```
aws firehose create-delivery-stream \
  --delivery-stream-name 'my-delivery-stream' \
  --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::222222222222:role/FirehoseToS3Role", "BucketARN":
  "arn:aws:s3:::amzn-s3-demo-bucket"}'
```

该输出值应该类似于以下内容：

```
{
  "DeliveryStreamARN": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream"
```

```
}
```

步骤 2：创建目标

Important

此过程中的所有步骤都需要在日志数据接收者账户中完成。

创建目标时，CloudWatch Logs 将代表收件人账户向目标发送测试消息。当订阅筛选条件稍后处于活动状态时，CloudWatch Logs 会代表源账户向目标发送录入事件。

创建目标

1. 请等待，直至您在 [步骤 1：创建 Firehose 传输流](#) 中创建的 Firehose 流变为活动状态。您可使用以下命令检查 StreamDescription.StreamStatus 属性。

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
```

此外，请记下 DeliveryStreamDescription.DeliveryStreamARN 值，因为后面的步骤中将需要用到它。此命令的示例输出：

```
{
  "DeliveryStreamDescription": {
    "DeliveryStreamName": "my-delivery-stream",
    "DeliveryStreamARN": "arn:aws:firehose:us-east-1:222222222222:deliverystream/my-delivery-stream",
    "DeliveryStreamStatus": "ACTIVE",
    "DeliveryStreamEncryptionConfiguration": {
      "Status": "DISABLED"
    },
    "DeliveryStreamType": "DirectPut",
    "VersionId": "1",
    "CreateTimestamp": "2021-02-01T23:59:15.567000-08:00",
    "Destinations": [
      {
        "DestinationId": "destinationId-000000000001",
        "S3DestinationDescription": {
          "RoleARN": "arn:aws:iam::222222222222:role/FirehoseToS3Role",
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
```

```

        "BufferingHints": {
            "SizeInMBs": 5,
            "IntervalInSeconds": 300
        },
        "CompressionFormat": "UNCOMPRESSED",
        "EncryptionConfiguration": {
            "NoEncryptionConfig": "NoEncryption"
        },
        "CloudWatchLoggingOptions": {
            "Enabled": false
        }
    },
    "ExtendedS3DestinationDescription": {
        "RoleARN": "arn:aws:iam::222222222222:role/FirehoseToS3Role",
        "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
        "BufferingHints": {
            "SizeInMBs": 5,
            "IntervalInSeconds": 300
        },
        "CompressionFormat": "UNCOMPRESSED",
        "EncryptionConfiguration": {
            "NoEncryptionConfig": "NoEncryption"
        },
        "CloudWatchLoggingOptions": {
            "Enabled": false
        },
        "S3BackupMode": "Disabled"
    }
},
    "HasMoreDestinations": false
}

```

您的传输流可能需要一两分钟才会显示为活动状态。

2. 当传输流处于活动状态时，请创建 IAM 角色，该角色将向 CloudWatch Logs 授予将数据放入 Firehose 流的权限。首先，您需要在文件 `~/TrustPolicyForCWL.json` 中创建信任策略。使用文本编辑器创建此策略。有关 CloudWatch Logs 端点的更多信息，请参阅 [Amazon CloudWatch Logs 端点和配额](#)。

此策略包括指定 `sourceAccountId` 的 `aws:SourceArn` 全局条件上下文密钥，有助于避免出现混淆代理安全问题。如果您在第一次调用中还不知道源账户 ID，则建议您将目标 ARN 放在源

ARN 字段中。在随后的调用中，应将源 ARN 设置为从第一次调用中收集的实际源 ARN。有关更多信息，请参阅 [混淆代理问题防范](#)。

```
{
  "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. 使用 `aws iam create-role` 命令创建 IAM 角色，并指定您刚创建的信任策略文件。

```
aws iam create-role \
  --role-name CWLtoKinesisFirehoseRole \
  --assume-role-policy-document file://~/TrustPolicyForCWL.json
```

以下内容为例输出。记下返回的 `Role.Arn` 值，因为您需要在后面的步骤中用到它。

```
{
  "Role": {
    "Path": "/",
    "RoleName": "CWLtoKinesisFirehoseRole",
    "RoleId": "AR0AR3BXASEKYJYWF243H",
    "Arn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
    "CreateDate": "2021-02-02T08:10:43+00:00",
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Effect": "Allow",
        "Principal": {
          "Service": "logs.region.amazonaws.com"
        },

```

```

        "Action": "sts:AssumeRole",
        "Condition": {
            "StringLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:region:sourceAccountId:*",
                    "arn:aws:logs:region:recipientAccountId:"
                ]
            }
        }
    }
}

```

4. 创建权限策略以定义 CloudWatch Logs 可对您的账户执行的操作。首先，使用文本编辑器在文件 `~/PermissionsForCWL.json` 中创建权限策略：

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["firehose:*"],
      "Resource": ["arn:aws:firehose:region:222222222222:*"]
    }
  ]
}

```

5. 通过输入以下命令，将权限策略与角色关联：

```

aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-name
Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json

```

6. 在 Firehose 传输流变为活动状态并且您已创建 IAM 角色后，便可以创建 CloudWatch Logs 目标。
 - a. 此步骤不会将访问策略与您的目标关联，它只是完成目标创建的两个步骤中的第一个步骤。记下有效负载中返回的新目标的 ARN，因为您将在后续步骤中使用它作为 `destination.arn`。

```

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. 上一步骤完成后，在日志数据接收者账户 (222222222222) 中将访问策略与目标关联。

此策略使得日志数据发送者账户 (111111111111) 可以仅访问日志数据接收者账户 (222222222222) 中的目标。您可以使用文本编辑器将此策略放在 `~/AccessPolicy.json` 文件中：

JSON

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111111111111"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}
```

- c. 这将创建一个策略，该策略定义了对目标具有写入权限的人。此策略必须指定 `logs:PutSubscriptionFilter` 操作才能访问目标。跨账户用户将使用 `PutSubscriptionFilter` 操作向目标发送日志事件：

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

步骤 3：添加/验证跨账户目标的 IAM 权限

根据 Amazon 跨账户策略评估逻辑，为访问任何跨账户资源（例如用作订阅筛选条件目标的 Kinesis 或 Firehose 流），您必须在发送账户中设置基于身份的策略，该策略提供对跨账户目标资源的明确访问权限。有关策略评估逻辑的更多信息，请参阅[跨账户策略评估逻辑](#)。

您可以将基于身份的策略附加到用于创建订阅筛选条件的 IAM 角色或 IAM 用户。此策略必须位于发送账户中。如果您使用管理员角色创建订阅筛选条件，则可以跳过此步骤继续前进至 [步骤 4：创建订阅筛选条件](#)。

添加或验证跨账户所需的 IAM 权限

1. 输入以下命令以检查使用哪个 IAM 角色或 IAM 用户来运行 Amazon 日志命令。

```
aws sts get-caller-identity
```

该命令返回的输出类似于下方内容：

```
{  
  "UserId": "User ID",  
  "Account": "sending account id",  
  "Arn": "arn:aws:sending account id:role/user:RoleName/UserName"  
}
```

记下由 *RoleName* 或 *UserName* 表示的值。

2. 登录发送账户中的 Amazon Web Services 管理控制台，使用您在步骤 1 所输入命令的输出中返回的 IAM 角色或 IAM 用户来搜索附加的策略。
3. 验证附加到该角色或用户的策略是否提供在跨账户目标资源上调用 `logs:PutSubscriptionFilter` 的明确权限。

以下策略提供权限，仅在单个 Amazon 账户（账户 123456789012）中对任何目标资源创建订阅筛选条件：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid":
"AllowSubscriptionFiltersOnAnyResourceInOneSpecificAccount",
      "Effect": "Allow",
      "Action": "logs:PutSubscriptionFilter",
      "Resource": [
        "arn:aws:logs:*:*:log-group:*",
        "arn:aws:logs*:123456789012:destination:*"
      ]
    }
  ]
}
```

以下策略提供权限，仅在单个 Amazon 账户（账户 123456789012）中对名为 sampleDestination 的特定目标资源创建订阅筛选条件：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSubscriptionFiltersOnSpecificResource",
      "Effect": "Allow",
      "Action": "logs:PutSubscriptionFilter",
      "Resource": [
        "arn:aws:logs:*:*:log-group:*",
        "arn:aws:logs*:123456789012:destination:amzn-s3-demo-bucket"
      ]
    }
  ]
}
```

步骤 4：创建订阅筛选条件

切换到发送账户，在此示例中为 111111111111。此时，您将在发送账户中创建订阅筛选条件。在此示例中，此筛选条件与一个包含 Amazon CloudTrail 事件的日志组关联，以便将“根”Amazon 凭证执行的每个记录下来的活动都传输到前面创建的目标。有关如何将 Amazon CloudTrail 事件发送到 CloudWatch Logs 的更多信息，请参阅 Amazon CloudTrail 用户指南中的[将 CloudTrail 事件发送到 CloudWatch Logs](#)。

输入以下命令时，请务必以 IAM 用户身份登录，或者使用在[步骤 3：添加/验证跨账户目标的 IAM 权限](#)中为其添加策略的 IAM 角色登录。

```
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"
```

日志组和目标必须位于同一 Amazon 区域内。但是，目标可指向位于不同区域的 Amazon 资源（如 Firehose 流）。

验证录入事件流

当您创建订阅筛选条件后，CloudWatch Logs 会将与筛选条件模式匹配的所有传入日志事件转发到 Firehose 传输流。根据 Firehose 传输流上设置的时间缓冲间隔，数据将开始显示在您的 Amazon S3 存储桶中。经过足够的时间后，您可以通过检查 Amazon S3 存储桶来验证您的数据。要检查存储桶，请输入以下命令：

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket'
```

该命令的输出将与以下内容类似：

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

然后，您可以通过输入以下命令，从存储桶中检索特定对象。将 key 的值替换为您在上一个命令中找到的值。

```
aws s3api get-object --bucket 'amzn-s3-demo-bucket' --key '2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba' testfile.gz
```

Amazon S3 对象中的数据以 gzip 格式压缩。您可使用以下命令之一检查命令行中的原始数据：

Linux：

```
zcat testfile.gz
```

macOS：

```
zcat <testfile.gz
```

在运行时修改目标成员资格

您可能会遇到必须在您拥有的目标中添加或删除日志发送者的情况。您可通过新访问策略对您的目标使用 PutDestinationPolicy 操作。在以下示例中，将阻止之前添加的账户 111111111111 再发送任何日志数据，并将启用账户 333333333333。

1. 提取当前与目标 testDestination 关联的策略并记下 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. 更新该策略，以体现已阻止账户 111111111111，并且账户 333333333333 已启用。将此策略放入 ~/NewAccessPolicy.json 文件：

JSON

```

{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "333333333333 "
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}

```

3. 使用以下命令，将 NewAccessPolicy.json 文件中定义的策略与目标关联：

```

aws logs put-destination-policy \
  --destination-name "testFirehoseDestination" \

  --access-policy file://~/NewAccessPolicy.json

```


这样就会最终禁用账户 ID 111111111111 中的日志事件。在账户 333333333333 的所有者创建订阅筛选条件后，账户 ID 333333333333 中的日志事件就会立即开始流向目标。

使用 Kinesis Data Streams 进行跨账户跨区域账户级订阅

创建跨账户订阅时，可以指定单个账户或企业作为发件人。如果指定企业，则此过程允许企业中的所有账户向接收方账户发送日志。

要跨账户共享日志数据，您需要建立日志数据发送者和接收者：

- Log data sender (日志数据发件人) – 从收件人处获取目标信息并告知 CloudWatch Logs 发件人已准备好将其录入事件发送到指定目标。在本部分其余程序中，日志数据发送者显示为虚构的 Amazon 账号 111111111111。

如果您将在企业中设立多个账户向收件人账户发送日志，则可以创建策略，授予企业中的所有账户向收件人账户发送日志的权限。您仍然必须为每个发件人账户设置单独的订阅筛选条件。

- 日志数据接收者 – 设置一个封装 Kinesis Data Streams 流的目标，并告知 CloudWatch Logs 接收者希望接收日志数据。接收者随后与发送者共享与此目标有关的信息。在本部分其余程序中，日志数据接收者显示为虚构的 Amazon 账号 999999999999。

要开始接收来自跨账户用户的日志事件，日志数据接收者应首先创建一个 CloudWatch Logs 目标。每个目标都包含以下键元素：

目标名称

您要创建的目标的名称。

目标 ARN

您要用作订阅源目标的 Amazon 资源的 Amazon Resource Name (ARN)。

角色 ARN

一个 Amazon Identity and Access Management (IAM) 角色，用于向 CloudWatch Logs 授予将数据放入所选流所需的必要权限。

访问策略

一个 IAM policy 文档（采用 JSON 格式，使用 IAM policy 语法编写），用于管理有权对您的目标进行写入的用户组。

 Note

日志组和目标必须位于同一Amazon区域内。但是，目标指向的Amazon资源可位于不同的区域。在以下部分的示例中，所有特定于区域的资源都是在美国东部（弗吉尼亚北部）创建的。

主题

- [设置新的跨账户订阅](#)
- [更新现有的跨账户订阅](#)

设置新的跨账户订阅

按照这些部分中所述步骤设置新的跨账户日志订阅。

主题

- [步骤 1：创建目标](#)
- [步骤 2：（仅在使用企业时）创建 IAM 角色](#)
- [步骤 3：创建账户级订阅筛选策略](#)
- [验证日志事件流](#)
- [在运行时修改目标成员资格](#)

步骤 1：创建目标

 Important

此过程中的所有步骤都需要在日志数据接收者账户中完成。

在本示例中，日志数据接收者账户的Amazon账户 ID 为 999999999999，而日志数据发送者Amazon账户 ID 为 111111111111。

该示例使用名为 RecipientStream 的 Kinesis Data Streams 流创建一个目标，并创建一个允许 CloudWatch Logs 将数据写入到该目标的角色。

创建目标时，CloudWatch Logs 将代表收件人账户向目标发送测试消息。当订阅筛选条件稍后处于活动状态时，CloudWatch Logs 会代表源账户向目标发送录入事件。

创建目标

1. 在收件人账户中，在 Kinesis Data Streams 中创建目标流。在命令提示符下，输入：

```
aws kinesis create-stream --stream-name "RecipientStream" --shard-count 1
```

2. 等到 流变为活动状态。您可使用 `aws kinesis describe-stream` 命令检查 `StreamDescription.StreamStatus` 属性。此外，请记下 `StreamDescription.StreamARN` 的值，因为稍后您需要将该值传递到 CloudWatch Logs：

```
aws kinesis describe-stream --stream-name "RecipientStream"
{
  "StreamDescription": {
    "StreamStatus": "ACTIVE",
    "StreamName": "RecipientStream",
    "StreamARN": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "34028236692093846346337460743176EXAMPLE",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
"4955113521868881845667950383198145878459135270218EXAMPLE"
        }
      }
    ]
  }
}
```

您的流可能需要一两分钟才会以活动状态显示。

3. 创建 IAM 角色，该角色将授予 CloudWatch Logs 将数据放入流的权限。首先，您需要在文件 `~/TrustPolicyForCWL.json` 中创建信任策略。使用文本编辑器创建此策略文件，请勿使用 IAM 控制台来创建。

此策略包括指定 `sourceAccountId` 的 `aws:SourceArn` 全局条件上下文密钥，有助于避免出现混淆代理安全问题。如果您在第一次调用中还不知道源账户 ID，则建议您将目标 ARN 放在源 ARN 字段中。在随后的调用中，应将源 ARN 设置为从第一次调用中收集的实际源 ARN。有关更多信息，请参阅 [混淆代理问题防范](#)。

```
{
  "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. 使用 `aws iam create-role` 命令创建 IAM 角色，并指定信任策略文件。记下返回的 `Role.Arn` 值，因为该值稍后也将传递到 CloudWatch Logs：

```
aws iam create-role \
--role-name CWLtoKinesisRole \
--assume-role-policy-document file://~/TrustPolicyForCWL.json

{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Condition": {
          "StringLike": {
            "aws:SourceArn": [
              "arn:aws:logs:region:sourceAccountId:",
              "arn:aws:logs:region:recipientAccountId:"
            ]
          }
        },
        "Principal": {
          "Service": "logs.amazonaws.com"
        }
      }
    }
  }
}
```

```

    }
  },
  "RoleId": "AA0IIAH450GAB4HC5F431",
  "CreateDate": "2023-05-29T13:46:29.431Z",
  "RoleName": "CWLtoKinesisRole",
  "Path": "/",
  "Arn": "arn:aws:iam::999999999999:role/CWLtoKinesisRole"
}
}

```

5. 创建权限策略以定义 CloudWatch Logs 可对您的账户执行的操作。首先，使用文本编辑器在文件 `~/PermissionsForCWL.json` 中创建权限策略：

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:999999999999:stream/RecipientStream"
    }
  ]
}

```

6. 使用 `aws iam put-role-policy` 命令将权限策略与角色关联：

```

aws iam put-role-policy \
  --role-name CWLtoKinesisRole \
  --policy-name Permissions-Policy-For-CWL \
  --policy-document file:///~/PermissionsForCWL.json

```

7. 在流变为活动状态并且您已创建 IAM 角色后，您便可以创建 CloudWatch Logs 目标。
 - a. 此步骤不会将访问策略与您的目标关联，它只是完成目标创建的两个步骤中的第一个步骤。记下负载中返回的 `DestinationArn`：

```

aws logs put-destination \
  --destination-name "testDestination" \
  --target-arn "arn:aws:kinesis:region:999999999999:stream/RecipientStream" \
  --role-arn "arn:aws:iam::999999999999:role/CWLtoKinesisRole"

{
  "DestinationName" : "testDestination",
  "RoleArn" : "arn:aws:iam::999999999999:role/CWLtoKinesisRole",

```

```

    "DestinationArn" : "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
    "TargetArn" : "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream"
  }

```

- b. 在步骤 7a 完成后，在日志数据接收者账户中将访问策略与目标关联。此策略必须指定 `logs:PutSubscriptionFilter` 操作并向发件人账户授予权限才能访问目标。

此策略向发送日志的 Amazon 账户授予权限。您可以在策略中仅指定这一个账户，如果发件人账户是企业的成员，则策略可以指定企业的企业 ID。这样，您只能创建策略，以允许企业中的多个账户向此目标账户发送日志。

使用文本编辑器创建名为 `~/AccessPolicy.json` 的文件，并包含下列策略语句之一。

第一个示例策略允许企业中所有 ID 为 `o-1234567890` 的账户将日志发送到收件人账户。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "logs:PutSubscriptionFilter",
        "logs:PutAccountPolicy"
      ],
      "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": [
            "o-1234567890"
          ]
        }
      }
    }
  ]
}

```

下一个示例只允许日志数据发件人账户 (111111111111) 将日志发送到日志数据收件人账户。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": [
        "logs:PutSubscriptionFilter",
        "logs:PutAccountPolicy"
      ],
      "Resource": "arn:aws:logs:us-  
east-1:999999999999:destination:testDestination"
    }
  ]
}
```

- c. 将您在上一步中创建的策略附加到目标。

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

此访问策略允许 ID 为 111111111111 的 Amazon 账户中的用户针对 ARN 为 `arn:aws:logs:region:999999999999:destination:testDestination` 的目标调用 `PutSubscriptionFilter`。任何其他用户针对此目标调用 `PutSubscriptionFilter` 的尝试都会被拒绝。

要针对访问策略验证用户的权限，请参阅 IAM 用户指南中的[使用策略验证程序](#)。

完成后，如果您为跨账户权限使用的是 Amazon Organizations，请按照 [步骤 2：（仅在使用企业时）创建 IAM 角色](#) 中的步骤操作。如果您选择直接向另一个账户授予权限而不是使用 Organizations，则可以跳过该步骤然后继续执行 [步骤 3：创建账户级订阅筛选策略](#)。

步骤 2：（仅在使用企业时）创建 IAM 角色

在上一节中，如果您使用向账户 111111111111 所在的企业授予权限的访问策略创建了目标，而不是直接向账户 111111111111 授予权限，则请按照本节中的步骤操作。否则，您可以跳至 [步骤 3：创建账户级订阅筛选策略](#)。

本节中的步骤创建了 IAM 角色，CloudWatch 可以代入该角色，并验证发件人账户是否有权针对收件人目标创建订阅筛选条件。

在发送者账户中执行本节中的步骤。该角色必须存在于发送者账户中，并且您在订阅筛选器中指定该角色的 ARN。在此示例中，发送者账户为 111111111111。

要使用 Amazon Organizations 为跨账户日志订阅创建所需的 IAM 角色

1. 请在文件 `/TrustPolicyForCWLSubscriptionFilter.json` 中创建以下信任策略。使用文本编辑器创建此策略文件；请勿使用 IAM 控制台来创建。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

2. 创建一个使用此策略的 IAM 角色。记录该命令返回的 `Arn` 值，您需要在本过程的后续部分中使用该值。在此示例中，我们将 `CWLtoSubscriptionFilterRole` 用作我们所创建角色的名称。

```
aws iam create-role \
  --role-name CWLtoSubscriptionFilterRole \
  --assume-role-policy-document file://~/
TrustPolicyForCWLSubscriptionFilter.json
```

3. 创建权限策略以定义 CloudWatch Logs 可对您的账户执行的操作。
 - a. 首先，使用文本编辑器在名为 `~/PermissionsForCWLSubscriptionFilter.json` 的文件中创建以下权限策略。


```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:PutLogEvents",
      "Resource": "arn:aws:logs:region:111111111111:log-group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
    }
  ]
}
```

- b. 输入以下命令，以将刚创建的权限策略与您在步骤 2 中创建的角色相关联。

```
aws iam put-role-policy
--role-name CWLtoSubscriptionFilterRole
--policy-name Permissions-Policy-For-CWL-Subscription-filter
--policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

完成后，您可以继续执行 [步骤 3：创建账户级订阅筛选策略](#)。

步骤 3：创建账户级订阅筛选策略

在创建目标后，日志数据接收者账户可与其他 Amazon 账户共享目标 ARN (arn:aws:logs:us-east-1:999999999999:destination:testDestination)，以便他们将日志事件发送到相同的目标。这些其他的发送账户用户随后会在各自的日志组上针对此目标创建订阅筛选条件。订阅筛选器将立即让实时日志数据开始从所选日志组向指定目标的流动。

Note

如果要向整个组织授予订阅筛选器的权限，则需要使用在 [步骤 2：（仅在使用企业时）创建 IAM 角色](#) 中创建的 IAM 角色的 ARN。

在以下示例中，在发送账户中创建了一个账户级订阅筛选条件。该筛选条件与发送者账户 111111111111 相关联，以便将与筛选条件和选择条件匹配的每个日志事件都传输到前面创建的目标。该目标封装一个名为“RecipientStream”的流。

selection-criteria 字段是可选的，但对于从订阅筛选条件中排除可能导致无限日志递归的日志组非常重要。有关此问题以及确定要排除哪些日志组的更多信息，请参阅 [防止日志递归](#)。目前，NOT IN 是 selection-criteria 唯一支持的运算符。

```
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"
```

发送者账户的日志组和目标必须位于同一 Amazon 区域内。但是，目标可指向位于不同区域的 Amazon 资源，如 Kinesis Data Streams 流。

验证日志事件流

在创建账户级订阅筛选策略后，CloudWatch Logs 会将与筛选条件模式和选择条件匹配的所有传入日志事件转发至封装在名为“RecipientStream”的目标流中的流。目标所有者可以使用 aws kinesis get-shard-iterator 命令抓取 Kinesis Data Streams 分片，并使用 aws kinesis get-records 命令提取一些 Kinesis Data Streams 记录以验证是否发生该过程：

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

{
  "ShardIterator":
  "AAAAAAAAAAAFGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+0IK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
}

aws kinesis get-records \
  --limit 10 \
  --shard-iterator
  "AAAAAAAAAAAFGU/
kLvNggvndHq2UIF0w5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
```

```
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3Igvq0b5dRA0yybNdRcdzvnC35KQANoHzzahKdRGB9v4scv+3vaq+f
+0IK8zM5My8ID+g6rMo7UKWeI4+IWIKEXAMPLE"
```

Note

您可能需要在 Kinesis Data Streams 开始返回数据前几分钟重新运行 `get-records` 命令。

您将看到包含一组 Kinesis Data Streams 记录的响应。Kinesis Data Streams 记录中的数据属性使用 gzip 格式压缩并采用 Base64 编码。您可以使用以下 Unix 命令检查命令行中的原始数据：

```
echo -n "<Content of Data>" | base64 -d | zcat
```

Base64 解码和解压缩数据被格式化为 JSON 并具有以下结构：

```
{
  "owner": "111111111111",
  "logGroup": "CloudTrail/logs",
  "logStream": "111111111111_CloudTrail/logs_us-east-1",
  "subscriptionFilters": [
    "RecipientStream"
  ],
  "messageType": "DATA_MESSAGE",
  "logEvents": [
    {
      "id": "3195310660696698337880902507980421114328961542429EXAMPLE",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
    }",
    },
    {
      "id": "3195310660696698337880902507980421114328961542429EXAMPLE",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
    }",
    },
    {
      "id": "3195310660696698337880902507980421114328961542429EXAMPLE",
      "timestamp": 1432826855000,
      "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
    }"
```

```
    }  
  ]  
}
```

数据架构中的关键元素如下：

messageType

数据消息将使用“DATA_MESSAGE”类型。有时候，CloudWatch Logs 可能发出类型为“CONTROL_MESSAGE”的 Kinesis Data Streams 记录，这主要是为了检查目标是否可访问。

owner

原始生成日志数据的 Amazon 账户 ID。

logGroup

原始日志数据的日志组名称。

logStream

原始日志数据的日志流名称。

subscriptionFilters

与原始日志数据匹配的订阅筛选条件名称的列表。

logEvents

表示为一组日志事件记录的实际日志数据。“id”属性是每个日志事件的唯一标识符。

policyLevel

强制实施策略的级别。“ACCOUNT_LEVEL_POLICY”是账户级订阅筛选策略的 policyLevel。

在运行时修改目标成员资格

您可能遇到必须在您拥有的目标中添加或删除某些用户的成员资格的情况。您可通过新访问策略对您的目标使用 `put-destination-policy` 命令。在以下示例中，将阻止之前添加的账户 111111111111 再发送任何日志数据，并将启用账户 222222222222。

1. 提取当前与目标 `testDestination` 关联的策略并记下 `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\" } ] }"
    }
  ]
}
```

- 更新该策略，以体现已阻止账户 111111111111，并且账户 222222222222 已启用。将此策略放入 ~/NewAccessPolicy.json 文件：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "222222222222"
      },
      "Action": [
        "logs:PutSubscriptionFilter",
        "logs:PutAccountPolicy"
      ],
      "Resource": "arn:aws:logs:us-east-1:999999999999:destination:testDestination"
    }
  ]
}
```

- 调用 PutDestinationPolicy 以将 NewAccessPolicy.json 文件中定义的策略与目标关联：

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

这将最终禁用账户 ID 111111111111 中的日志事件。在账户 222222222222 的所有者创建订阅筛选条件后，账户 ID 222222222222 中的日志事件就会立即开始流向目标。

更新现有的跨账户订阅

如果您当前有跨账户日志订阅，其中目标账户仅向特定发件人账户授予权限，并且您想更新此订阅以便目标账户向企业中所有账户授予访问权限，请按照本节中的步骤操作。

主题

- [步骤 1：更新订阅筛选条件](#)
- [步骤 2：更新现有的目标访问策略](#)

步骤 1：更新订阅筛选条件

Note

此步骤仅对于由 [启用从 Amazon 服务进行日志记录](#) 中列出的服务创建的日志的跨账户订阅才需要。如果您不使用由这些日志组之一创建的日志，则可以跳至 [步骤 2：更新现有的目标访问策略](#)。

在某些情况下，您必须更新向目标账户发送日志的所有发件人账户中的订阅筛选条件。此更新添加了 IAM 角色，CloudWatch 可以承担该角色，并验证发件人账户是否有向收件人账户发送日志的权限。

请按照本节中的步骤为您要更新的每个发件人账户执行操作，以使用企业 ID 获得跨账户订阅权限。

在本节的示例中，111111111111 和 222222222222 两个账户已经创建了用于向账户 999999999999 发送日志的订阅筛选条件。现有的订阅筛选条件值如下：

```
## Existing Subscription Filter parameter values  
{  
  "DestinationArn": "arn:aws:logs:region:999999999999:destination:testDestination",  
  "FilterPattern": "{$.userIdentity.type = Root}",  
  "Distribution": "Random"
```

```
}
```

如果需要查找当前订阅筛选条件参数值，请输入以下命令。

```
aws logs describe-account-policies \  
--policy-type "SUBSCRIPTION_FILTER_POLICY" \  
--policy-name "CrossAccountStreamsExamplePolicy"
```

要更新订阅筛选条件以开始使用企业 ID 获取跨账户日志权限

1. 请在文件 `~/TrustPolicyForCWL.json` 中创建以下信任策略。使用文本编辑器创建此策略文件；请勿使用 IAM 控制台来创建。

```
{  
  "Statement": {  
    "Effect": "Allow",  
    "Principal": { "Service": "logs.amazonaws.com" },  
    "Action": "sts:AssumeRole"  
  }  
}
```

2. 创建一个使用此策略的 IAM 角色。记录该命令返回的 Arn 值的 Arn 值，您将需要在本过程的后部分中使用该值。在此示例中，我们将 `CWLtoSubscriptionFilterRole` 用作我们所创建角色的名称。

```
aws iam create-role  
  \ --role-name CWLtoSubscriptionFilterRole  
  \ --assume-role-policy-document file://~/TrustPolicyForCWL.json
```

3. 创建权限策略以定义 CloudWatch Logs 可对您的账户执行的操作。
 - a. 首先，使用文本编辑器在名为 `/PermissionsForCWLSubscriptionFilter.json` 的文件中创建以下权限策略。

```
{  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "logs:PutLogEvents",  
      "Resource": "arn:aws:logs:region:111111111111:log-  
group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
```

```
    }
  ]
}
```

- b. 输入以下命令，以将刚创建的权限策略与您在步骤 2 中创建的角色相关联。

```
aws iam put-role-policy
  --role-name CWLtoSubscriptionFilterRole
  --policy-name Permissions-Policy-For-CWL-Subscription-filter
  --policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

4. 输入以下命令以更新订阅筛选策略。

```
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"
```

步骤 2：更新现有的目标访问策略

更新所有发件人账户中的订阅筛选条件后，可以更新收件人账户中的目标访问策略。

在以下示例中，收件人账户为 999999999999，目的地名为 testDestination。

此更新将使企业中 ID 为 o-1234567890 的所有账户向收件人账户发送日志。只有创建了订阅筛选条件的账户才会实际向收件人账户发送日志。

要更新收件人账户中的目标访问策略以开始使用企业 ID 获取权限

1. 在收件人账户中，请使用文本编辑器创建包含下列内容的 ~/AccessPolicy.json 文件。

JSON

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



```
{
  "Sid": "",
  "Effect": "Allow",
  "Principal": "*",
  "Action": [
    "logs:PutSubscriptionFilter",
    "logs:PutAccountPolicy"
  ],
  "Resource": "arn:aws:logs:us-east-1:999999999999:destination:testDestination",
  "Condition": {
    "StringEquals": {
      "aws:PrincipalOrgID": [
        "o-1234567890"
      ]
    }
  }
}
```

2. 输入以下命令，将刚创建的策略附加到现有目标。要更新目标以使用具有企业 ID 的访问策略而不是列出特定 Amazon 账户 ID 的访问策略，请包括 force 参数。

 Warning

如果您正在使用由 Amazon 中列出的 [启用从 Amazon 服务进行日志记录](#) 服务发送的日志，则在执行此步骤之前，您必须先按照 [步骤 1：更新订阅筛选条件](#) 中的说明更新所有发件人账户中的订阅筛选条件。

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

使用 Firehose 进行跨账户跨区域账户级订阅

要跨账户共享日志数据，您需要建立日志数据发送者和接收者：

- 日志数据发送者 – 从接收者处获取目标信息并告知 CloudWatch Logs 发送者已准备好将其日志事件发送到指定目标。在本部分其余程序中，日志数据发送者显示为虚构的 Amazon 账号 111111111111。
- 日志数据接收者 – 设置一个封装 Kinesis Data Streams 流的目标，并告知 CloudWatch Logs 接收者希望接收日志数据。接收者随后与发送者共享与此目标有关的信息。在本部分其余程序中，日志数据接收者显示为虚构的 Amazon 账号 222222222222。

本节中的示例使用带有 Amazon S3 存储的 Firehose 传输流。您还可以使用不同的设置来设置 Firehose 传输流。有关更多信息，请参阅 [Creating a Firehose Delivery Stream](#)。

Note

日志组和目标必须位于同一 Amazon 区域内。但是，目标指向的 Amazon 资源可位于不同的区域。

Note

支持同一账户和跨区域传输流的 Firehose 订阅筛选条件。

主题

- [步骤 1：创建 Firehose 传输流](#)
- [步骤 2：创建目标](#)
- [步骤 3：创建账户级订阅筛选策略](#)
- [验证录入事件流](#)
- [在运行时修改目标成员资格](#)

步骤 1：创建 Firehose 传输流

Important

在完成以下步骤之前，您必须使用访问策略，以便 Firehose 可以访问您的 Amazon S3 存储桶。有关更多信息，请参阅《Amazon Data Firehose 开发人员指南》中的 [Controlling Access](#)。

本部分（步骤 1）中的所有步骤都需要在日志数据接收者账户中完成。

以下示例命令中使用的是美国东部（弗吉尼亚州北部）。请将此区域替换为适用于您的部署的正确区域。

创建将用作目标的 Firehose 传输流

1. 创建 Amazon S3 存储桶：

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --create-bucket-configuration
LocationConstraint=us-east-1
```

2. 创建 IAM 角色，授予 Firehose 将数据放入存储桶的权限。

a. 首先，使用文本编辑器在文件 `~/TrustPolicyForFirehose.json` 中创建信任策略。

```
{ "Statement": { "Effect": "Allow", "Principal": { "Service":
  "firehose.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition":
  { "StringEquals": { "sts:ExternalId": "222222222222" } } } }
```

b. 创建 IAM 角色，并指定您刚创建的信任策略文件。

```
aws iam create-role \
  --role-name FirehoseToS3Role \
  --assume-role-policy-document file:///~/TrustPolicyForFirehose.json
```

c. 此命令的输出与以下内容类似。记下角色名称和角色 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. 创建权限策略，以定义 Firehose 可对您的账户执行的操作。

- a. 首先，使用文本编辑器在名为 `~/PermissionsForFirehose.json` 的文件中创建以下权限策略。根据应用场景，您可能需要为此文件添加更多权限。

```

{
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:ListBucket"
    ],
    "Resource": [
      "arn:aws:s3:::amzn-s3-demo-bucket",
      "arn:aws:s3:::amzn-s3-demo-bucket/*"
    ]
  }]
}

```

- b. 输入以下命令，以将刚创建的权限策略与 IAM 角色相关联。

```

aws iam put-role-policy --role-name FirehoseToS3Role --policy-name
Permissions-Policy-For-Firehose-To-S3 --policy-document file://~/
PermissionsForFirehose.json

```

4. 输入以下命令以创建 Firehose 传输流。将 *my-role-arn* 和 *amzn-s3-demo-bucket2-arn* 替换为正确的部署值。

```

aws firehose create-delivery-stream \
  --delivery-stream-name 'my-delivery-stream' \
  --s3-destination-configuration \

```

```
'{"RoleARN": "arn:aws:iam::222222222222:role/FirehoseToS3Role", "BucketARN":  
"arn:aws:s3:::amzn-s3-demo-bucket"}'
```

该输出值应该类似于以下内容：

```
{  
  "DeliveryStreamARN": "arn:aws:firehose:us-east-1:222222222222:deliverystream/  
my-delivery-stream"  
}
```

步骤 2：创建目标

Important

此过程中的所有步骤都需要在日志数据接收者账户中完成。

创建目标时，CloudWatch Logs 将代表收件人账户向目标发送测试消息。当订阅筛选条件稍后处于活动状态时，CloudWatch Logs 会代表源账户向目标发送录入事件。

创建目标

1. 请等待，直至您在 [步骤 1：创建 Firehose 传输流](#) 中创建的 Firehose 流变为活动状态。您可使用以下命令检查 StreamDescription.StreamStatus 属性。

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
```

此外，请记下 DeliveryStreamDescription.DeliveryStreamARN 值，因为后面的步骤中将需要用到它。此命令的示例输出：

```
{  
  "DeliveryStreamDescription": {  
    "DeliveryStreamName": "my-delivery-stream",  
    "DeliveryStreamARN": "arn:aws:firehose:us-  
east-1:222222222222:deliverystream/my-delivery-stream",  
    "DeliveryStreamStatus": "ACTIVE",  
    "DeliveryStreamEncryptionConfiguration": {  
      "Status": "DISABLED"  
    },  
  },  
}
```

```

    "DeliveryStreamType": "DirectPut",
    "VersionId": "1",
    "CreateTimestamp": "2021-02-01T23:59:15.567000-08:00",
    "Destinations": [
      {
        "DestinationId": "destinationId-000000000001",
        "S3DestinationDescription": {
          "RoleARN": "arn:aws:iam::222222222222:role/FirehoseToS3Role",
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
          "BufferingHints": {
            "SizeInMBs": 5,
            "IntervalInSeconds": 300
          },
          "CompressionFormat": "UNCOMPRESSED",
          "EncryptionConfiguration": {
            "NoEncryptionConfig": "NoEncryption"
          },
          "CloudWatchLoggingOptions": {
            "Enabled": false
          }
        },
        "ExtendedS3DestinationDescription": {
          "RoleARN": "arn:aws:iam::222222222222:role/FirehoseToS3Role",
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
          "BufferingHints": {
            "SizeInMBs": 5,
            "IntervalInSeconds": 300
          },
          "CompressionFormat": "UNCOMPRESSED",
          "EncryptionConfiguration": {
            "NoEncryptionConfig": "NoEncryption"
          },
          "CloudWatchLoggingOptions": {
            "Enabled": false
          },
          "S3BackupMode": "Disabled"
        }
      }
    ],
    "HasMoreDestinations": false
  }
}

```

您的传输流可能需要一两分钟才会显示为活动状态。

2. 当传输流处于活动状态时，请创建 IAM 角色，该角色将向 CloudWatch Logs 授予将数据放入 Firehose 流的权限。首先，您需要在文件 `~/TrustPolicyForCWL.json` 中创建信任策略。使用文本编辑器创建此策略。有关 CloudWatch Logs 端点的更多信息，请参阅 [Amazon CloudWatch Logs 端点和配额](#)。

此策略包括指定 `sourceAccountId` 的 `aws:SourceArn` 全局条件上下文密钥，有助于避免出现混淆代理安全问题。如果您在第一次调用中还不知道源账户 ID，则建议您将目标 ARN 放在源 ARN 字段中。在随后的调用中，应将源 ARN 设置为从第一次调用中收集的实际源 ARN。有关更多信息，请参阅 [混淆代理问题防范](#)。

```
{
  "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. 使用 `aws iam create-role` 命令创建 IAM 角色，并指定您刚创建的信任策略文件。

```
aws iam create-role \
  --role-name CWLtoKinesisFirehoseRole \
  --assume-role-policy-document file://~/TrustPolicyForCWL.json
```

以下内容为示例输出。记下返回的 `Role.Arn` 值，因为您需要在后面的步骤中用到它。

```
{
  "Role": {
    "Path": "/",
```

```

    "RoleName": "CWLtoKinesisFirehoseRole",
    "RoleId": "AR0AR3BXASEKYJYWF243H",
    "Arn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
    "CreateDate": "2023-02-02T08:10:43+00:00",
    "AssumeRolePolicyDocument": {
      "Statement": {
        "Effect": "Allow",
        "Principal": {
          "Service": "logs.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
          "StringLike": {
            "aws:SourceArn": [
              "arn:aws:logs:region:sourceAccountId:*",
              "arn:aws:logs:region:recipientAccountId:"
            ]
          }
        }
      }
    }
  }
}

```

4. 创建权限策略以定义 CloudWatch Logs 可对您的账户执行的操作。首先，使用文本编辑器在文件 `~/PermissionsForCWL.json` 中创建权限策略：

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["firehose:*"],
      "Resource": ["arn:aws:firehose:region:222222222222:*"]
    }
  ]
}

```

5. 通过输入以下命令，将权限策略与角色关联：

```

aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-name
Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json

```


6. 在 Firehose 传输流变为活动状态并且您已创建 IAM 角色后，便可以创建 CloudWatch Logs 目标。
 - a. 此步骤不会将访问策略与您的目标关联，它只是完成目标创建的两个步骤中的第一个步骤。记下有效负载中返回的新目标的 ARN，因为您将在后续步骤中使用它作为 `destination.arn`。

```
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. 上一步骤完成后，在日志数据接收者账户 (222222222222) 中将访问策略与目标关联。此策略使得日志数据发送者账户 (111111111111) 可以仅访问日志数据接收者账户 (222222222222) 中的目标。您可以使用文本编辑器将此策略放在 `~/AccessPolicy.json` 文件中：

JSON

```
{  
    "Version": "2012-10-17",  
    "Statement" : [  
        {  
            "Sid" : "",  
            "Effect" : "Allow",  
            "Principal" : {  
                "AWS" : "111111111111"  
            },  
            "Action" : ["logs:PutSubscriptionFilter", "logs:PutAccountPolicy"],  
        }  
    ]  
}
```

```
"Resource" : "arn:aws:logs:us-east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}
```

- c. 这将创建一个策略，该策略定义了对目标具有写入权限的人。此策略必须指定 `logs:PutSubscriptionFilter` 和 `logs:PutAccountPolicy` 操作才能访问目标。跨账户用户将使用 `PutSubscriptionFilter` 和 `PutAccountPolicy` 操作向目标发送日志事件。

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

步骤 3：创建账户级订阅筛选策略

切换到发送账户，在此示例中为 111111111111。您现在将在发送账户中创建账户级订阅筛选策略。在此示例中，筛选条件会导致除两个日志组之外的所有日志组中包含字符串 `ERROR` 的每个日志事件都传输到前面创建的目标位置。

```
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"
```

发送账户的日志组和目标必须位于同一 Amazon 区域内。但是，目标可指向位于不同区域的 Amazon 资源（如 Firehose 流）。

验证录入事件流

当您创建订阅筛选条件后，CloudWatch Logs 会将与筛选条件模式和选择条件匹配的所有传入日志事件转发到 Firehose 传输流。根据 Firehose 传输流上设置的时间缓冲间隔，数据将开始显示在您的 Amazon S3 存储桶中。经过足够的时间后，您可以通过检查 Amazon S3 存储桶来验证您的数据。要检查存储桶，请输入以下命令：

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket'
```

该命令的输出将与以下内容类似：

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

然后，您可以通过输入以下命令，从存储桶中检索特定对象。将 key 的值替换为您在上一个命令中找到的值。

```
aws s3api get-object --bucket 'amzn-s3-demo-bucket' --key '2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba' testfile.gz
```

Amazon S3 对象中的数据以 gzip 格式压缩。您可使用以下命令之一检查命令行中的原始数据：

Linux：

```
zcat testfile.gz
```

macOS：

```
zcat <testfile.gz
```

在运行时修改目标成员资格

您可能会遇到必须在您拥有的目标中添加或删除日志发送者的情况。您可通过新访问策略对您的目标使用 `PutDestinationPolicy` 和 `PutAccountPolicy` 操作。在以下示例中，将阻止之前添加的账户 111111111111 再发送任何日志数据，并将启用账户 333333333333。

1. 提取当前与目标 `testDestination` 关联的策略并记下 `AccessPolicy`：

```
aws logs describe-destinations \
  --destination-name-prefix "testFirehoseDestination"
```

返回的数据可能如下所示。

```
{
  "destinations": [
    {
      "destinationName": "testFirehoseDestination",
      "targetArn": "arn:aws:firehose:us-east-1:222222222222:deliverystream/my-delivery-stream",
      "roleArn": "arn:aws:iam:: 222222222222:role/CWLtoKinesisFirehoseRole",
      "accessPolicy": "{\n  \"Version\" : \"2012-10-17\",\n  \"Statement\" : [\n    {\n      \"Sid\" : \"\",\n      \"Effect\" : \"Allow\",\n      \"Principal\" : {\n        \"AWS\" : \"111111111111\"\n      },\n      \"Action\" : \"logs:PutSubscriptionFilter\",\n      \"Resource\" : \"arn:aws:logs:us-east-1:222222222222:destination:testFirehoseDestination\"\n    }\n  ]\n}",
      "arn": "arn:aws:logs:us-east-1:222222222222:destination:testFirehoseDestination",
      "creationTime": 1612256124430
    }
  ]
}
```

2. 更新该策略，以体现已阻止账户 111111111111，并且账户 333333333333 已启用。将此策略放入 `~/NewAccessPolicy.json` 文件：

JSON

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

```
"Sid" : "",
"Effect" : "Allow",
"Principal" : {
  "AWS" : "333333333333 "
},
"Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
"Resource" : "arn:aws:logs:us-east-1:222222222222:destination:testFirehoseDestination"
}
]
```

3. 使用以下命令，将 NewAccessPolicy.json 文件中定义的策略与目标关联：

```
aws logs put-destination-policy \
  --destination-name "testFirehoseDestination" \

  --access-policy file://~/NewAccessPolicy.json
```

这样就会最终禁用账户 ID 111111111111 中的日志事件。在账户 333333333333 的所有者创建订阅筛选条件后，账户 ID 333333333333 中的日志事件就会立即开始流向目标。

混淆代理问题防范

混淆代理问题是一个安全性问题，即不具有某操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在 Amazon 中，跨服务模拟可能会导致混淆代理问题。一个服务（呼叫服务）调用另一项服务（所谓的“服务”）时，可能会发生跨服务模拟。可以操纵调用服务，使用其权限以在其他情况下该服务不应有访问权限的方式对另一个客户的资源进行操作。为防止这种情况，Amazon 提供可帮助您保护所有服务的数据的工具，而这些服务中的服务主体有权限访问账户中的资源。

我们建议在资源策略中使用 [aws:SourceArn](#)、[aws:SourceAccount](#)、[aws:SourceOrgID](#) 和 [aws:SourceOrgPaths](#) 全局条件上下文键，以限制为其他服务提供的资源访问权限。使用 [aws:SourceArn](#) 来仅将一个资源与跨服务访问相关联。使用 [aws:SourceAccount](#) 来让该账户中的任何资源与跨服务使用相关联。使用 [aws:SourceOrgID](#) 来允许某组织内的任何账户中的任何资源与跨服务使用相关联。使用 [aws:SourceOrgPaths](#) 来将 Amazon Organizations 路径内账户中的任何资源与跨服务使用相关联。有关使用和了解路径的更多信息，请参阅[了解 Amazon Organizations 实体路径](#)。

防范混淆代理问题最有效的方法是使用 `aws:SourceArn` 全局条件上下文键和资源的完整 ARN。如果不知道资源的完整 ARN，或者正在指定多个资源，请针对 ARN 未知部分使用带有通配符字符 (*) 的 `aws:SourceArn` 全局上下文条件键。例如 `arn:aws:service:*:123456789012:*`。

如果 `aws:SourceArn` 值不包含账户 ID，例如 Amazon S3 桶 ARN，您必须使用 `aws:SourceAccount` 和 `aws:SourceArn` 来限制权限。

要防范大规模混淆代理问题，请在基于资源的策略中将 `aws:SourceOrgID` 或 `aws:SourceOrgPaths` 全局条件上下文键与资源的组织 ID 或组织路径一起使用。包含 `aws:SourceOrgID` 或 `aws:SourceOrgPaths` 键的策略将自动包含正确的账户，并且当您在组织中添加、删除或移动账户时，无需手动更新策略。

[步骤 1：创建目标](#)和[步骤 2：创建目标](#)中所述的用于向 CloudWatch Logs 授予将数据写入 Kinesis Data Streams 和 Firehose 的权限的策略展示了如何使用 `aws:SourceArn` 全局条件上下文键来帮助防止混淆代理问题。

防止日志递归

订阅筛选条件存在导致无限日志递归的风险，如果不加以阻止，则可能会导致 CloudWatch Logs 和目标中的提取计费大幅增加。当订阅筛选条件与由于订阅传输工作流而接收日志事件的日志组关联时，会发生这种情况。提取到日志组中的日志将传输到目标位置，从而导致日志组提取更多日志，然后这些日志将再次转发到目标位置，从而形成递归循环。

例如，考虑一个目标位置为 Firehose 的订阅筛选条件，它将日志事件传输到 Amazon S3。此外，还有一个 Lambda 函数，可以处理传输到 Amazon S3 的新事件并自行生成一些日志。如果将订阅筛选条件应用于 Lambda 函数的日志组，则该函数生成的日志事件将转发到目标位置的 Firehose 和 Amazon S3，然后它们将再次调用该函数，从而生成更多日志并将其转发到 Firehose 和 Amazon S3，导致再次调用该函数等。这将陷入无限循环，导致日志提取、Firehose 和 Amazon S3 的计费意外增加。

如果 Lambda 函数附加到为 CloudWatch Logs 启用了流日志的 VPC，则 VPC 的日志组也可能导致日志递归。

我们建议您不要将订阅筛选条件应用于属于订阅传输工作流一部分的日志组。对于账户级订阅筛选条件，请使用 `PutAccountPolicy` API 中的 `selectionCriteria` 参数从策略中排除这些日志组。

排除日志组时，请考虑以下生成日志并可能属于订阅传输工作流一部分的 Amazon 服务：

- 带有 Fargate 的 Amazon EC2
- Lambda

- Amazon Step Functions
- 为 CloudWatch Logs 启用的 Amazon VPC 流日志

 Note

Lambda 目标的日志组生成的日志事件将不会被转发回账户级订阅筛选策略的 Lambda 函数。在这种情况下，账户订阅策略不需要使用 `selectionCriteria` 排除目标 Lambda 函数的日志组。

指标筛选条件、订阅筛选条件、筛选日志事件和 Live Tail 的筛选条件模式语法

Note

有关如何使用 Amazon CloudWatch Logs Insights 查询语言查询日志组的信息，请参阅 [CloudWatch Logs Insights 语法查询语法](#)。

借助 CloudWatch Logs，您可以使用[指标筛选条件](#)将日志数据转换为可操作的指标，使用[订阅筛选条件](#)将日志事件路由到其他 Amazon 服务，使用[筛选日志事件](#)来搜索日志事件，以及使用 [Live Tail](#) 实时交互式查看提取的日志。

筛选模式构成了指标筛选条件、订阅筛选条件、日志事件和 Live Tail 用来匹配日志事件中的字词的语法。字词可以是单词、准确的短语或数字值。正则表达式 (regex) 可用于创建独立的筛选条件模式，也可以与 JSON 和以空格分隔的筛选条件模式合并。

使用要匹配的字词创建筛选条件模式。筛选条件模式仅返回包含您定义的字词的录入事件。您可以在 CloudWatch 控制台中测试筛选条件模式。

主题

- [支持的正则表达式 \(regex \) 语法](#)
- [使用筛选条件模式将字词与正则表达式 \(regex \) 进行匹配](#)
- [使用筛选条件模式匹配非结构化日志事件中的字词](#)
- [使用筛选条件模式匹配 JSON 日志事件中的字词](#)
- [使用筛选条件模式来匹配以空格分隔的日志事件中的字词](#)

支持的正则表达式 (regex) 语法

支持的正则表达式语法

使用正则表达式搜索和筛选日志数据时，必须用 % 将表达式括起来。

使用正则表达式的筛选条件模式只能包括以下内容：

- 字母数字字符 – 字母数字字符是字母 (从 A 到 Z 或 a 到 z) 或数字 (从 0 到 9) 字符。

- 支持的符号字符 – 这些符号包括：“:”、“_”、“#”、“=”、“@”、“/”、“;”、“,”和“-”。例如，%something!% 会被拒绝，因为不支持“!”。
- 支持的运算符 – 这些运算符包括：“^”、“\$”、“?”、“[”、“]”、“{”、“}”、“|”、“\”、“*”、“+”和“.”。

不支持 (和) 运算符。不能使用括号来定义子模式。

不支持多字节字符。

Note

限额

在创建指标筛选条件或订阅筛选条件时，每个日志组最多有 5 个包含正则表达式的筛选条件模式。

为指标筛选条件和订阅筛选条件创建带分隔符的筛选条件模式或 JSON 筛选条件模式，或者筛选日志事件或 Live Tail 时，每个筛选条件模式的正则表达式限制为 2 个。

支持的运算符的使用

- ^：将匹配项锚定到字符串的开头。例如，%^[hc]at% 匹配“hat”和“cat”，但仅在字符串的开头。
- \$：将匹配项锚定到字符串的结尾。例如， %[hc]at\$% 匹配“hat”和“cat”，但仅在字符串的结尾。
- ?：匹配前面字词的零个或多个出现次数。例如， %colou?r% 可以同时匹配“color”和“colour”。
- []：定义字符类。匹配方括号内包含的字符列表或字符范围。例如， %[abc]% 匹配“a”、“b”或“c”； %[a-z]% 匹配从“a”到“z”的任何小写字母； %[abcx-z]% 匹配“a”、“b”、“c”、“x”、“y”或“z”。
- {m, n}：与前面的字词匹配至少 m 次且不超过 n 次。例如， %a{3,5}% 仅匹配“aaa”、“aaaa”和“aaaaa”。

Note

如果您选择不定义最小值或最大值，则可以省略 m 或 n。

- |：布尔值“Or”，与竖线两侧的字词相匹配。例如：
 - %gra|ey% 可以匹配“gray”或“grey”
 - %^starting|^initializing|^shutting down% 可以匹配“starting ...”或“initializing ...”或“shutting down”，但与“kipping initializing ...”不匹配

- `%abcc|ab[^c]$` 可以匹配“abcc ...”和“aba ...”，但不匹配“aac ...”
- `\`：转义字符，允许您使用运算符的字面含义而不是其特殊含义。例如，`%\[.\]%` 匹配任何由“[”和“]”括起来的单个字符，因为方括号已转义，例如“[a]”、“[b]”、“[7]”、“[@]”、“[]”和“[]”。

 Note

`%10\.10\.0\.1%` 是创建正则表达式来匹配 IP 地址 10.10.0.1 的正确方法。

- `*`：匹配前面字词的零个或多个实例。例如，`%ab*c%` 可以匹配“ac”、“abc”和“abbbc”；`%ab[0-9]*%` 可以匹配“ab”、“ab0”和“ab129”。
- `+`：匹配前面字词的一个或多个实例。例如，`%ab+c%` 可以匹配“abc”、“abbc”和“abbbc”，但不匹配“ac”。
- `.`：匹配任意单个字符。例如，`%.at%` 匹配以“at”结尾的任意三个字符串，包括“hat”、“cat”、“bat”、“4at”、“#at”和“ at”（以空格开头）。

 Note

在创建正则表达式以匹配 IP 地址时，转义 `.` 运算符很重要。例如，`%10.10.0.1%` 可能匹配“10010,051”，这可能不是表达式的实际预期用途。

- `\d`、`\D`：匹配数字/非数字字符。例如，`%\d%` 等同于 `%[0-9]%`，`%\D%` 等同于 `%[^0-9]%`。

 Note

大写运算符表示其小写对应运算符的逆运算符。

- `\s`、`\S`：匹配空格字符/非空格字符。

 Note

大写运算符表示其小写对应运算符的逆运算符。空格字符包括制表符 (`\t`)、空格 () 和换行符 (`\n`)。

- `\w`、`\W`：匹配字母数字字符/非字母数字字符。例如，`%\w%` 等同于 `%[a-zA-Z_0-9]%`，`%\W%` 等同于 `%[^a-zA-Z_0-9]%`。

Note

大写运算符表示其小写对应运算符的逆运算符。

- `\xhh`：匹配两位十六进制字符的 ASCII 映射。`\x` 是转义序列，表示以下字符代表 ASCII 的十六进制值。hh 指定指向 ASCII 表中字符的两个十六进制数字（0-9 和 A-F）。

Note

您可以使用 `\xhh` 匹配筛选条件模式不支持的符号字符。例如，`%\x3A%` 匹配 `;`；`%\x28%` 匹配 `(`。

使用筛选条件模式将字词与正则表达式（regex）进行匹配

使用正则表达式匹配字词

您可以使用 `%` 括起来的正则表达式模式（正则表达式模式前后有百分号）来匹配日志事件中的字词。下面的代码段显示了筛选条件模式的示例，该模式返回包含 `AUTHORIZED` 关键字的所有日志事件。

有关支持的正则表达式的列表，请参阅[支持的正则表达式](#)。

```
%AUTHORIZED%
```

此筛选条件模式返回日志事件消息，例如以下内容：

- `[ERROR 401] UNAUTHORIZED REQUEST`
- `[SUCCESS 200] AUTHORIZED REQUEST`

使用筛选条件模式匹配非结构化日志事件中的字词

匹配非结构化日志事件中的字词

以下示例包含代码段，这些代码段显示了如何使用筛选条件模式匹配非结构化日志事件中的字词。

 Note

筛选条件模式区分大小写。将包含非字母数字字符的精确短语和字词用双引号括起来 (")。

Example: Match a single term

下面的代码段显示了单个字词筛选条件模式的示例，该模式返回其中的消息包含字词 ERROR 的所有录入事件。

```
ERROR
```

此筛选条件模式匹配日志事件消息，例如以下内容：

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST
- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

Example: Match multiple terms

下面的代码段显示了多个字词筛选条件模式的示例，该模式返回其中的消息包含字词 ERROR 和 ARGUMENTS 的所有录入事件。

```
ERROR ARGUMENTS
```

筛选条件返回录入事件消息，例如以下内容：

- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

筛选条件模式不会返回以下日志事件消息，因为它们不包含筛选条件模式中指定的两个字词。

- [ERROR 400] BAD REQUEST

- [ERROR 401] UNAUTHORIZED REQUEST

Example: Match optional terms

您可以使用模式匹配来创建返回包含可选字词的日志事件的筛选条件模式。在您想匹配的字词前放置问号 ("?")。下面的代码片段将显示筛选条件模式的一个示例，它将返回消息中包含单词 ERROR 或 ARGUMENTS 的所有日志事件。

```
?ERROR ?ARGUMENTS
```

此筛选条件模式匹配日志事件消息，例如以下内容：

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST
- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

Note

不能将问号 (“?”) 与其他筛选条件模式组合，例如包含和排除术语。如果将 “?” 与其他筛选模式结合使用，则所有带问号的字词都将被忽略。

例如，以下筛选模式匹配所有包含词语 REQUEST 的事件，但带有问号 (“?”) 的筛选词将被忽略且无效。

```
?ERROR ?ARGUMENTS REQUEST
```

日志事件匹配项

- [INFO] REQUEST FAILED
- [WARN] UNAUTHORIZED REQUEST
- [ERROR] 400 BAD REQUEST

Example: Match exact phrases

下面的代码段显示了筛选条件模式的示例，该模式返回其中的消息包含精确短语 INTERNAL SERVER ERROR 的录入事件。

```
"INTERNAL SERVER ERROR"
```

此筛选条件模式返回以下日志事件消息：

- [ERROR 500] INTERNAL SERVER ERROR

Example: Include and exclude terms

您可以创建筛选条件模式，该模式返回录入事件，其中的消息包含一些字词并排除其他字词。在您想排除的字词前放置减号 ("-")。下面的代码段显示了筛选条件模式的示例，该模式返回其中的消息包含字词 ERROR 并排除字词 ARGUMENTS 的录入事件。

```
ERROR -ARGUMENTS
```

此筛选条件模式返回日志事件消息，例如以下内容：

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST

此筛选条件模式不会返回以下日志事件消息，因为它们包含单词 ARGUMENTS。

- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

Example: Match everything

您可以用双引号匹配录入事件中的所有内容。下面的代码段显示了筛选条件模式的示例，该模式返回所有录入事件。

```
" "
```

使用筛选条件模式匹配 JSON 日志事件中的字词

为 JSON 日志事件编写筛选条件模式

下面介绍了如何编写与包含字符串和数值的 JSON 字词匹配的筛选条件模式的语法。

Writing filter patterns that match strings

您可以创建筛选条件模式以匹配 JSON 日志事件中的字符串。以下代码段显示了基于字符串的筛选条件模式的语法示例。

```
{ PropertySelector EqualityOperator String }
```

用大括号 ("{}") 括住筛选条件模式。基于字符串的筛选条件模式必须包含以下部分：

- 属性选择器

用美元符号后跟句点 ("\$.") 来启动属性选择器。属性选择器是字母数字字符串，它还支持连字符 ("-") 和下划线 ("_") 字符。字符串不支持科学记数法。属性选择器指向 JSON 录入事件中的值节点。值节点可以是字符串或数字。将数组放在属性选择器之后。数组中的元素遵循从零开始的编号系统，即数组中的第一个元素是元素 0，第二个元素是元素 1，依此类推。将元素括在括号 ("[]") 中。如果属性选择器指向数组或对象，则筛选条件模式与日志格式将不匹配。如果 JSON 属性包含句点 (".".")，则可以使用括号表示法来选择该属性。

Note

通配符选择器

您可以使用 JSON 通配符来选择任何数组元素或任何 JSON 对象字段。

限额

在一个属性选择器中最多只能使用一个通配符选择器。

- 等号运算符

使用以下符号之一来启动等号运算符：等于 ("=") 或不等于 ("!=")。等号运算符返回布尔值 (true 或 false)。

- 字符串

您可以用双引号 ("") 将字符串括起来。包含除字母数字字符和下划线符号以外的其他类型的字符串必须放在双引号中。使用星号 ("*") 作为通配符来匹配文本。

 Note

创建筛选条件模式来匹配 JSON 日志事件中的字词时，可以使用任何条件正则表达式。有关支持的正则表达式的列表，请参阅[支持的正则表达式](#)。

以下代码段包含筛选条件模式的示例，显示了如何格式化筛选条件模式，以将 JSON 字词与字符串匹配。

```
{ $.eventType = "UpdateTrail" }
```

Writing filter patterns that match numeric values

您可以创建筛选条件模式以匹配 JSON 日志事件中的数字值。以下代码段显示了与数字值匹配的筛选条件模式的语法示例。

```
{ PropertySelector NumericOperator Number }
```

用大括号 ("{} ") 括住筛选条件模式。与数字值匹配的筛选条件模式必须包含以下部分：

- 属性选择器

用美元符号后跟句点 ("\$.") 来启动属性选择器。属性选择器是字母数字字符串，它还支持连字符 ("-") 和下划线 ("_") 字符。字符串不支持科学记数法。属性选择器指向 JSON 录入事件中的值节点。值节点可以是字符串或数字。将数组放在属性选择器之后。数组中的元素遵循从零开始的编号系统，即数组中的第一个元素是元素 0，第二个元素是元素 1，依此类推。将元素括在括号 ("[]") 中。如果属性选择器指向数组或对象，则筛选条件模式与日志格式将不匹配。如果 JSON 属性包含句点 (".".")，则可以使用括号表示法来选择该属性。

 Note

通配符选择器

您可以使用 JSON 通配符来选择任何数组元素或任何 JSON 对象字段。

限额

在一个属性选择器中最多只能使用一个通配符选择器。

- 数值运算符

使用以下符号之一来启动数字运算符：大于 (">")、小于 ("<")、等于 ("=")、不等于 ("!=")、大于等于 (">=") 或小于等于 ("<=")。

- Number[]

您可以使用包含加号 ("+") 或减号 ("-") 符号的整数并遵循科学记数法。使用星号 ("*") 作为通配符来匹配数字。

以下代码段包含示例，显示了如何格式化筛选条件模式，以将 JSON 字词与数字值匹配。

```
// 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 }
```

使用简单表达式匹配 JSON 日志事件中的字词

以下示例包含代码段，显示了筛选条件模式如何与 JSON 日志事件中的字词匹配。

 Note

如果您使用示例 JSON 日志事件测试示例筛选条件模式，则必须在单行中输入示例 JSON 日志。

JSON 日志事件

```
{
  "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

此筛选条件模式与属性 "eventType" 中的字符串 "UpdateTrail" 匹配。

```
{ $.eventType = "UpdateTrail" }
```

Example: Filter pattern that matches string values (IP address)

此筛选条件模式包含通配符并与属性 "sourceIPAddress" 匹配，因为它不包含带有前缀 "123.123." 的数字。

```
{ $.sourceIPAddress != 123.123.* }
```

Example: Filter pattern that matches a specific array element with a string value

此筛选条件模式与数组 "arrayKey" 中的元素 "value" 匹配。

```
{ $.arrayKey[0] = "value" }
```

Example: Filter pattern that matches a string using regex

此筛选条件模式与属性 "eventType" 中的字符串 "Trail" 匹配。

```
{ $.eventType = %Trail% }
```

Example: Filter pattern that uses a wildcard to match values of any element in the array using regex

筛选条件模式包含与数组 "arrayKey" 中的元素 "value" 匹配的正则表达式。

```
{ $.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)

此筛选条件模式包含与属性 "sourceIPAddress" 中的元素 "111.111.111.111" 匹配的正则表达式。

```
{ $.* = %111\.111\.111\.1[0-9]{1,2}% }
```

 Note

限额

在一个属性选择器中最多只能使用一个通配符选择器。

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

您可以创建筛选条件模式，它们使用 IS 变量匹配 JSON 日志中的字段。IS 变量可以匹配包含值 NULL、TRUE 或 FALSE 的字段。下面的筛选条件模式返回 JSON 日志，其中 SomeObject 的值为 NULL。

```
{ $.SomeObject IS NULL }
```

Example: Filter pattern that matches JSON logs using NOT EXISTS

您可以使用 NOT EXISTS 变量创建筛选模式，以返回日志数据中不包含特定字段的 JSON 日志。下面的筛选条件模式使用 NOT EXISTS 返回不包含字段 SomeOtherObject 的 JSON 日志。

```
{ $.SomeOtherObject NOT EXISTS }
```

Note

目前不支持变量 IS NOT 和 EXISTS。

使用复合表达式匹配 JSON 对象中的字词

您可以在筛选条件模式中使用逻辑运算符 AND (“&&”) 和 OR (“||”) 来创建匹配两个或更多条件为 true 的日志事件的复合表达式。复合表达式支持使用括号 (“()”) 和以下运算的标准顺序：() > && > ||。以下示例包含代码段，这些代码段显示了如何使用具有复合表达式的筛选条件模式来匹配 JSON 对象中的字词。

JSON 对象

```
{
  "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 (&&)

此筛选条件模式包含复合表达式，该表达式匹配 "user" 中的 "id" (数字值为 1) 和 "users" 数组中第一个元素中的 "email" (字符串为 "John.Doe@example.com") 。

```
{ ($.user.id = 1) && ($.users[0].email = "John.Doe@example.com") }
```

Example: Expression that matches using OR (||)

此筛选条件模式包含复合表达式，匹配 "user" 中的 "email" (字符串为 "John.Stiles@example.com") 。

```
{ $.user.email = "John.Stiles@example.com" || $.coordinates[0][1] = "nonmatch" && $.actions[2] = "nonmatch" }
```

Example: Expression that doesn't match using AND (&&)

此筛选条件模式包含无法找到匹配项的复合表达式，因为该表达式与 "actions" 中的第三个操作不匹配。

```
{ ($.user.email = "John.Stiles@example.com" || $.coordinates[0][1] = "nonmatch") && $.actions[2] = "nonmatch" }
```

Note

限额

在属性选择器中最多只能使用一个通配符选择器，在具有复合表达式的筛选条件模式中最多只能使用三个通配符选择器。

Example: Expression that doesn't match using OR (||)

此筛选条件模式包含无法找到匹配项的复合表达式，因为该表达式与 "users" 中的第一个属性不匹配，或 "actions" 中的第三个操作不匹配。

```
{ ($.user.id = 2 && $.users[0].email = "nonmatch") || $.actions[2] = "GET" }
```

使用筛选条件模式来匹配以空格分隔的日志事件中的字词

为以空格分隔的日志事件编写筛选条件模式

您可以创建筛选条件模式来匹配以空格分隔的日志事件中的字词。下面提供了一个以空格分隔的日志事件示例，并介绍了如何编写与以空格分隔的日志事件中的字词匹配的筛选条件模式语法。

Note

创建筛选条件模式来匹配以空格分隔的日志事件中的字词时，可以使用任何条件正则表达式。有关支持的正则表达式的列表，请参阅[支持的正则表达式](#)。

Example: Space-delimited log event

以下代码段显示了以空格分隔的录入事件，其中包含七个字段：ip、user、username、timestamp、request、status_code 和 bytes。

```
127.0.0.1 Prod frank [10/Oct/2000:13:25:15 -0700] "GET /index.html HTTP/1.0" 404 1534
```

Note

括号 ("[]") 和双引号 ("") 之间的字符被视为单个字段。

Writing filter patterns that match terms in a space-delimited log event

要创建与以空格分隔的日志事件中的字词匹配的筛选条件模式，请用方括号 ("[]") 将筛选条件模式括起来，并以不同名称指定各个字段，用逗号 (",") 隔开。以下筛选条件模式解析七个字段。

```
[ip=%127\.\0\.\0\.[1-9]%, user, username, timestamp, request =*.html*, status_code = 4*, bytes]
```

您可以使用数字运算符 (>、<、=、!=、>= 或 <=) 和星号 (*) 作为通配符或正则表达式，用于为筛选条件模式提供条件。在示例筛选条件模式中，ip 使用与 IP 地址范围 127.0.0.1 – 127.0.0.9 匹配的正则表达式，request 包含通配符，表示它必须提取以 .html 开头的值，status_code 包含通配符，表示它必须提取以 4 开头的值。

如果您不知道在以空格分隔的录入事件中解析的字段数量，则可以使用省略号 (...) 来引用任何未命名的字段。省略号可以根据需要引用尽可能多的字段。以下示例显示带省略号的筛选条件模式，该筛选条件表示之前的示例筛选条件模式中显示的前四个未命名字段。

```
[..., request =*.html*, status_code = 4*, bytes]
```

您还可以使用逻辑运算符 AND (&&) 和 OR (||) 来创建复合表达式。以下筛选条件模式包含复合表达式，该表达式表示 status_code 的值必须是 404 或 410。

```
[ip, user, username, timestamp, request =*.html*, status_code = 404 || status_code = 410, bytes]
```

使用模式匹配来匹配以空格分隔的日志事件中的字词

您可以使用模式匹配来创建以空格分隔的筛选条件模式，以匹配特定顺序的字词。使用指示符指定字词的顺序。使用 w1 代表您的第一个字词，以及 w2 以此类推，来表示后续字词的顺序。在字词之间放置逗号 (",")。以下示例包含代码段，这些代码段显示了如何将模式匹配与以空格分隔的筛选条件模式结合使用。

Note

创建筛选条件模式来匹配以空格分隔的日志事件中的字词时，可以使用任何条件正则表达式。有关支持的正则表达式的列表，请参阅[支持的正则表达式](#)。

以空格分隔的日志事件


```
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

以下以空格分隔的筛选条件模式会返回日志事件，其中事件中的第一个单词是 ERROR。

```
[w1=ERROR, w2]
```

Note

创建使用模式匹配的以空格分隔的筛选条件模式时，必须在指定字词顺序后包含空白指示符。例如，如果您创建了一个筛选模式，其返回第一个单词为 ERROR 的日志事件，请在 w1 字词后包括空白 w2 指示符。

Example: Match terms with AND (&&) and OR (||)

您可以使用逻辑运算符 AND (“&&”) 和 OR (“||”) 创建包含条件的以空格分隔的筛选条件模式。以下筛选条件模式会返回日志事件，其中事件中的第一个单词是 ERROR 或 WARNING。

```
[w1=ERROR || w1=WARNING, w2]
```

Example: Exclude terms from matches

您可以创建以空格分隔的筛选条件模式，该筛选条件模式会返回排除一个或多个字词的日志事件。在您想排除的字词前放置不等号 (“!=”)。以下代码段显示了筛选条件模式的示例，该筛选条件模式返回第一个字词不是 ERROR 和 WARNING 的日志事件。

```
[w1!=ERROR && w1!=WARNING, w2]
```

Example: Match the top level item in a resource URI

以下代码段显示了筛选条件模式的示例，该筛选条件模式使用正则表达式匹配资源 URI 中的顶级项目。

```
[logLevel, date, time, method, url=%/service/resource/[0-9]+$, response_time]
```

Example: Match the child level item in a resource URI

以下代码段显示了筛选条件模式的示例，该筛选条件模式使用正则表达式匹配资源 URI 中的子级项目。

```
[logLevel, date, time, method, url=%/service/resource/[0-9]+/part/[0-9]+$,  
response_time]
```

启用从 Amazon 服务进行日志记录

虽然许多服务仅将日志发布到 CloudWatch Logs，但有些Amazon服务可以将日志直接发布到 Amazon Simple Storage Service 或 Amazon Data Firehose。如果您对日志的主要需求是在其中一种服务中存储或处理日志，那么您可以轻松地让生成日志的服务将它们直接发送到 Amazon S3 或 Firehose，而无需额外设置。

即使日志直接发布到 Amazon S3 或 Firehose，仍会产生费用。有关更多信息，请参阅 [Amazon CloudWatch Pricing \(Amazon CloudWatch 定价 \)](#)，Logs (日志) 选项卡上的 Vended Logs (已出售日志)。

一些 Amazon 服务使用通用基础设施来发送日志。要启用从这些服务进行日志记录，您必须以具有特定权限的用户身份登录。此外，您必须向 Amazon 授予权限才能发送日志。

对于需要这些权限的服务，所需的权限有两个版本。表中将需要这些额外权限的服务标为支持的 [V1 权限]和支持的 [V2 权限]。有关这些必需权限的信息，请参阅表后的部分。

日志源	日志类型	the section called “发送到 CloudWatch Logs 的日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “跟踪发送到 X-Ray”
Amazon API Gateway 访问日志	Vended logs (已出售日志)	支持的 [V1 权限]			
Amazon AppSync 日志	Custom logs (自定义日志)	支持			
Amazon Aurora MySQL 日志	Custom logs (自	支持			

日志源	日志类型	the section called “发送到 CloudWatch Logs 的日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “跟踪发送到 X-Ray”
	定义日志)				
Amazon Bedrock 知识库日志记录	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Bedrock 代理日志记录	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Bedrock AgentCore 运行时	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Bedrock AgentCore 网关	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Bedrock AgentCore 身份	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]

日志源	日志类型	the section called “发送到 CloudWatch Logs 的日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “跟踪发送到 X-Ray”
Amazon Bedrock AgentCore 工具	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Chime 媒体质量指标日志和 SIP 消息日志	Vended logs (已出售日志)	支持的 [V1 权限]			
CloudFront : 访问日志	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon CloudHSM 审计日志	Custom logs (自定义日志)	支持			
CloudWatch Evidently 评估事件日志	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]		
CloudWatch Internet Monitor 日志	Vended logs (已出售日志)		支持的 [V1 权限]		

日志源	日志类型	the section called “发送到 CloudWatch Logs 的日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “跟踪发送到 X-Ray”
CloudTrail 日志	Custom logs (自定义日志)	支持			
Amazon CodeBuild 日志	Custom logs (自定义日志)	支持			
Amazon CodeWhisperer 事件日志	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Cognito 日志	Vended logs (已出售日志)	支持的 [V1 权限]			
Amazon Connect 日志	Custom logs (自定义日志)	支持			
Amazon DataSync 日志	Custom logs (自定义日志)	支持			

日志源	日志类型	the section called “发送到 CloudWatch Logs 的日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “跟踪发送到 X-Ray”
Amazon ElastiCache (Redis OSS) 日志	Vended logs (已出售日志)	支持的 [V1 权限]		支持的 [V1 权限]	
Amazon Elastic Beanstalk 日志	Custom logs (自定义日志)	支持			
Amazon Elastic Container Service 日志	Custom logs (自定义日志)	支持			
Amazon Elastic Kubernetes Service 控制面板日志	Vended logs (已出售日志)	支持			
AWS Elemental MediaPackage 访问日志	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
AWS Elemental MediaTailor 日志	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]

日志源	日志类型	the section called “发送到 CloudWatch Logs 的日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “跟踪发送到 X-Ray”
Amazon Entity Resolution 数据匹配服务 日志	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon EventBridge 管道日志记录	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]	
Amazon EventBridge 事件总线	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Fargate 日志	Custom logs (自定义日志)	支持			
Amazon Fault Injection Service 实验日志	Vended logs (已出售日志)		支持的 [V1 权限]		
Amazon FinSpace	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]	

日志源	日志类型	the section called “发送到 CloudWatch Logs 的日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “跟踪发送到 X-Ray”
Amazon Global Accelerator 流日志	Vended logs (已售日志)		支持的 [V1 权限]		
Amazon Glue 作业日志	Custom logs (自定义日志)	支持			
IAM Identity Center 错误日志	Vended logs (已售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Interactive Video Service 聊天日志	Vended logs (已售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]	
Amazon IoT 日志	Custom logs (自定义日志)	支持			
Amazon IoT FleetWise 日志	Vended logs (已售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]	

日志源	日志类型	the section called “发送到 CloudWatch Logs 的日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “跟踪发送到 X-Ray”
Amazon Lambda 日志	Vended logs (已出售日志)	支持	支持	支持	
Amazon Macie 日志	Custom logs (自定义日志)	支持			
Amazon SES 日志	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Mainframe Modernization	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]	
Amazon Managed Service for Prometheus 日志	Vended logs (已出售日志)	支持的 [V1 权限]			
Amazon MSK 代理日志	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]	

日志源	日志类型	the section called “发送到 CloudWatch Logs 的日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “跟踪发送到 X-Ray”
Amazon MSK Connect 日志	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]	
Amazon MQ 日志	Custom logs (自定义日志)	支持			
Amazon 网络防火墙日志	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]	
Network Load Balancer 访问日志	Vended logs (已出售日志)		支持的 [V1 权限]		
OpenSearch 日志	Custom logs (自定义日志)	支持			
Amazon OpenSearch Service 摄取日志	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]	

日志源	日志类型	the section called “发送到 CloudWatch Logs 的日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “跟踪发送到 X-Ray”
Amazon PCS 日志	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Relational Database ServicePostgreSQL 日志	Custom logs (自定义日志)	支持			
Amazon RTB Fabric 日志	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Q Business 连接器日志	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Q Business 对话日志	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Route 53 公有 DNS 查询日志	Vended logs (已出售日志)	支持			

日志源	日志类型	the section called “发送到 CloudWatch Logs 的日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “跟踪发送到 X-Ray”
– Amazon Route 53 Resolver 查询日志	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]		
Amazon SageMaker AI 事件	Vended logs (已出售日志)	支持的 [V1 权限]			
Amazon SageMaker AI 工作线程事件	Vended logs (已出售日志)	支持的 [V1 权限]			
Amazon Site-to-Site VPN 日志	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]	
Amazon Simple Email Service 日志	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Simple Notification Service 日志	Custom logs (自定义日志)	支持			

日志源	日志类型	the section called “发送到 CloudWatch Logs 的日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “跟踪发送到 X-Ray”
Amazon Simple Notification Service 数据保护策略日志	Custom logs (自定义日志)	支持			
EC2 Spot 实例数据源文件	Vended logs (已出售日志)		支持的 [V1 权限]		
Amazon Step Functions 表达工作流和标准工作流日志	Vended logs (已出售日志)	支持的 [V1 权限]			
Storage Gateway 审计日志和运行状况日志	Vended logs (已出售日志)	支持的 [V1 权限]			
Amazon Transfer Family 日志	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]	
Amazon Verified Access 日志	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]	

日志源	日志类型	the section called “发送到 CloudWatch Logs 的日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “跟踪发送到 X-Ray”
Amazon Virtual Private Cloud 流日志	Vended logs (已出售日志)	支持	支持的 [V1 权限]	支持的 [V1 权限]	
Amazon VPC Lattice 访问日志	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]	
Amazon VPC Route Server	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon WAF 日志	Vended logs (已出售日志)	支持的 [V1 权限]	支持的 [V1 权限]	支持	
Amazon WorkMail 审计日志	Vended logs (已出售日志)	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]

需要额外权限 [V1] 的日志记录

一些 Amazon 服务使用通用基础设施将其日志发送到 CloudWatch Logs、Amazon S3 或 Firehose。要启用下表中列出的 Amazon 服务，将其日志发送到前述目标，您必须以具有特定权限的用户身份登录。

此外，必须将权限授予Amazon以启用日志发送。Amazon可以在设置日志时自动创建这些权限，您还可以在设置日志记录之前自行创建这些权限。对于跨账户传输，您必须自己手动创建权限策略。

如果您或您企业中的某个人在首次设置日志发送时选择让Amazon自动设置必要的权限和资源策略，那么设置日志发送的用户必须具有一定的权限，如本节后面所述。或者，您可以自行创建资源策略，这样设置日志发送的用户就不需要那么多权限。

以下主题提供了每个目标的更多详细信息。

主题

- [发送到 CloudWatch Logs 的日志](#)
- [发送到 Amazon S3 的日志](#)
- [日志发送至 Firehose](#)

发送到 CloudWatch Logs 的日志

Important

当您将以下列表中的日志类型设置为发送到 CloudWatch Logs 时，Amazon会创建或更改与接收日志的日志组关联的资源策略（如需要）。继续阅读本节，查看详细信息。

当前一部分的表中列出的日志类型发送到 CloudWatch Logs 时，本部分适用：

用户权限

若要能够设置首次将这些类型日志中的任一种日志发送到 CloudWatch Logs，您必须登录到具有以下权限的账户。

- `logs:CreateLogDelivery`
- `logs:PutResourcePolicy`
- `logs:DescribeResourcePolicies`
- `logs:DescribeLogGroups`

Note

当您指定 `logs:DescribeLogGroups`、`logs:DescribeResourcePolicies` 或 `logs:PutResourcePolicy` 权限时，请确保将其 Resource 行的 ARN 设置为使用 *

通配符，而不是仅指定单个日志组名称。例如，"Resource": "arn:aws:logs:us-east-1:111122223333:log-group:*"

如果其中任何一种类型的日志已被发送到 CloudWatch Logs 中的日志组，要设置将其中另一种日志也发送到同一日志组，您只需要 logs:CreateLogDelivery 权限。

日志组和资源策略

接收日志的日志组必须具有包含特定权限的资源策略。如果日志组当前没有资源策略，而且设置日志记录的用户对日志组具有 logs:PutResourcePolicy、logs:DescribeResourcePolicies 和 logs:DescribeLogGroups 权限，那么当您开始将日志发送到 CloudWatch Logs 日志时，Amazon 会自动为其创建以下策略。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite20150319",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "delivery.logs.amazonaws.com"
        ]
      },
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group:log-
stream:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": [
            "0123456789"
          ]
        }
      }
    }
  ]
}
```

```
    "ArnLike": {
      "aws:SourceArn": [
        "arn:aws:logs:us-east-1:111122223333:*"
      ]
    }
  }
}
```

如果日志组具有资源策略，但该策略不包含上一个策略中所示的语句，并且设置日志记录的用户对日志组具有 `logs:PutResourcePolicy`、`logs:DescribeResourcePolicies` 和 `logs:DescribeLogGroups` 权限，则该语句将附加到日志组的资源策略中。

发送到 Amazon S3 的日志

当您将日志设置为发送到 Amazon S3 时，Amazon 会创建或更改与接收日志的 S3 存储桶关联的资源策略（如需要）。

直接发布到 Amazon S3 的日志将发布到您指定的现有存储桶。每 5 分钟将在指定的存储桶中创建一个或多个日志文件。

当您首次将日志发送到 Amazon S3 存储桶时，发送日志的服务会记录存储桶的拥有者，以确保日志仅发送到属于该账户的存储桶。因此，要更改 Amazon S3 存储桶拥有者，您必须在原始服务中重新创建或更新日志订阅。

Note

CloudFront 使用与将公开发布的日志发送到 S3 的其他服务不同的权限模型。有关更多信息，请参阅[配置标准日志记录和访问日志文件所需的权限](#)。

此外，如果您对 CloudFront 访问日志和另一个日志源使用相同的 S3 存储桶，则在 CloudFront 的存储桶上启用 ACL 也会向使用此存储桶的所有其他日志源授予权限。

Important

如果您要将日志发送到 Amazon S3 存储桶，并且存储桶策略包含 `NotAction` 或 `NotPrincipal` 元素，则自动向存储桶添加日志传输权限和创建日志订阅将会失败。为了成功

创建日志订阅，您需要手动将日志传输权限添加到存储桶策略，然后创建日志订阅。有关更多信息，请参阅本节中的说明。

如果存储桶使用客户自主管理型 Amazon KMS 密钥进行服务器端加密，您还必须为客户自主管理型密钥添加密钥政策。有关更多信息，请参阅 [Amazon S3](#)。

如果目标存储桶启用了 SSE-KMS 和存储桶密钥，则附加的客户管理型 KMS 密钥策略将不再按预期对所有请求工作。有关更多信息，请参阅[使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本](#)。

如果您使用公开发布的日志和具有客户管理型 Amazon KMS 密钥的 S3 加密，则在配置存储桶时，您必须使用完全限定的 Amazon KMS 密钥 ARN 而不是密钥 ID。有关更多信息，请参阅[put-bucket-encryption](#)。

用户权限

若要能够设置首次将这些类型日志中的任一种日志发送到 Amazon S3，您必须登录具有以下权限的账户。

- logs:CreateLogDelivery
- S3:GetBucketPolicy
- S3:PutBucketPolicy

如果其中任何一种类型的日志已被发送到 Amazon S3 存储桶，要设置将其中另一种日志也发送到同一存储桶，您只需要 logs:CreateLogDelivery 权限。

S3 存储桶资源策略

接收日志的 S3 存储桶必须具有包含特定权限的资源策略。如果存储桶当前没有资源策略，而且设置日志记录的用户对存储桶具有 S3:GetBucketPolicy 和 S3:PutBucketPolicy 权限，那么当您开始将日志发送到 Amazon S3 时，Amazon 会自动为其创建以下策略。

JSON

```
{
  "Version": "2012-10-17",
  "Id": "AWSLogDeliveryWrite20150319",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryAclCheck",
```

```

        "Effect": "Allow",
        "Principal": {
            "Service": "delivery.logs.amazonaws.com"
        },
        "Action": "s3:GetBucketAcl",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": [
                    "0123456789"
                ]
            },
            "ArnLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:us-east-1:111122223333:*"
                ]
            }
        }
    },
    {
        "Sid": "AWSLogDeliveryWrite",
        "Effect": "Allow",
        "Principal": {
            "Service": "delivery.logs.amazonaws.com"
        },
        "Action": "s3:PutObject",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/AWSLogs/account-ID/*",
        "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control",
                "aws:SourceAccount": [
                    "0123456789"
                ]
            },
            "ArnLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:us-east-1:111122223333:*"
                ]
            }
        }
    }
]
}

```

在之前的策略中，对于 `aws:SourceAccount`，请指定将日志传送到此存储桶的账户 ID 列表。对于 `aws:SourceArn`，请按 `arn:aws:logs:source-region:source-account-id:*` 格式指定生成日志的资源 ARN 列表。

如果存储桶具有资源策略，但该策略不包含上一个策略中所示的语句，并且设置日志记录的用户对存储桶具有 `S3:GetBucketPolicy` 和 `S3:PutBucketPolicy` 权限，则该语句将附加到存储桶的资源策略中。

Note

某些情况下，如果未向 `delivery.logs.amazonaws.com` 授予 `s3:ListBucket` 权限，可能会在 Amazon CloudTrail 中看到 `AccessDenied` 错误。为了避免 CloudTrail 日志中出现此类错误，必须向 `delivery.logs.amazonaws.com` 授予 `s3:ListBucket` 权限且必须包含与在上述存储桶策略中设置的 `s3:GetBucketAcl` 权限一起显示的 `Condition` 参数。为方便起见，可以直接将 `AWSLogDeliveryAclCheck` 更新为 `"Action": ["s3:GetBucketAcl", "s3:ListBucket"]`，而不是创建一个新的 `Statement`

Amazon S3 存储桶服务器端加密

您可以通过启用 Amazon S3 托管式密钥 (SSE-S3) 的服务器端加密或 Amazon Key Management Service 中存储 Amazon KMS 密钥 (SSE-KMS) 的服务器端加密来保护 Amazon S3 存储桶中的数据。有关更多信息，请参阅[使用服务器端加密保护数据](#)。

如果选择 SSE-S3，则不需要额外的配置。Amazon S3 处理加密密钥。

Warning

如果选择 SSE-KMS，则必须使用客户托管式密钥，因为此场景不支持使用 Amazon 托管式密钥。如果您使用 Amazon 托管密钥设置加密，则会以不可读取的格式提供日志。

当您使用客户托管式 Amazon KMS 密钥时，您可以在启用存储桶加密时指定客户托管式密钥的 Amazon Resource Name (ARN)。您必须将以下内容添加到客户托管式密钥的密钥策略（不是 S3 存储桶的存储桶策略）中，以便日志传输账户可以写入 S3 存储桶。

如果选择 SSE-KMS，则必须使用客户托管式密钥，因为此场景不支持使用 Amazon 托管式密钥。当您使用客户托管式 Amazon KMS 密钥时，您可以在启用存储桶加密时指定客户托管式密钥的 Amazon

Resource Name (ARN)。您必须将以下内容添加到客户托管式密钥的密钥策略 (不是 S3 存储桶的存储桶策略) 中, 以便日志传输账户可以写入 S3 存储桶。

```
{
  "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:*"]
    }
  }
}
```

对于 `aws:SourceAccount`, 请指定将日志传送到此存储桶的账户 ID 列表。对于 `aws:SourceArn`, 请按 `arn:aws:logs:source-region:source-account-id:*` 格式指定生成日志的资源 ARN 列表。

日志发送至 Firehose

当前一部分的表中列出的日志类型发送到 Firehose 时, 本部分适用:

用户权限

若要能够设置首次将这些类型日志中的任一种日志发送到 Firehose, 您必须登录具有以下权限的账户。

- `logs:CreateLogDelivery`
- `firehose:TagDeliveryStream`

- iam:CreateServiceLinkedRole

如果其中任何一种日志已被发送到 Firehose，要设置将其中另一种日志也发送到 Firehose，您只需要 logs:CreateLogDelivery 和 firehose:TagDeliveryStream 权限。

用于权限的 IAM 角色

由于 Firehose 不使用资源策略，Amazon 将这些日志设置为发送到 Firehose 时会使用 IAM 角色。Amazon 创建名为 AWSServiceRoleForLogDelivery 的服务相关角色。此服务相关角色包括以下权限。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "firehose:PutRecord",
        "firehose:PutRecordBatch",
        "firehose:ListTagsForDeliveryStream"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/LogDeliveryEnabled": "true"
        }
      },
      "Effect": "Allow"
    }
  ]
}
```

此服务相关角色会为将 LogDeliveryEnabled 标签设置为 true 的所有 Firehose 传输流授予权限。当您设置日志记录时，Amazon 会将此标签提供给目标传输流。

此服务相关角色还具有允许 delivery.logs.amazonaws.com 服务委托人来代入所需服务相关角色的信任策略。该信任策略如下所示：

JSON

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

需要额外权限 [V2] 的日志记录

一些 Amazon 服务使用新的方法来发送日志。这是一种灵活的方法，可让您设置从这些服务向以下一个或多个目标传输日志：CloudWatch Logs、Amazon S3 或 Firehose 和 X-Ray（用于跟踪传输）。

工作日志传输由三个元素组成：

- **DeliverySource**，这是代表实际发送日志的资源的逻辑对象。
- **DeliveryDestination**，这是代表实际传输目标的逻辑对象。
- **Delivery**，这连接传输源与传输目标

要在支持的 Amazon 服务和目标之间配置日志传输，您必须执行以下操作：

- 使用 [putDeliverySource](#) 创建传输源。
- 使用 [putDeliveryDestination](#) 创建传输目标。
- 如果要跨账户传输日志，则必须在目标账户中使用 [PutDeliveryDestinationPolicy](#) 为目标分配 IAM 策略。此策略授权创建从账户 A 中的传输源到账户 B 中的传输目标的传输。对于跨账户传输，您必须手动创建权限策略。
- 通过使用 [CreateDelivery](#) 将一个传输源和一个传输目标精确配对来创建传输。

以下各节详细介绍了在登录期间使用 V2 流程将日志传输到每种类型目标时需要具备的权限。这些权限可以授予您登录时使用的 IAM 角色。

Important

删除日志生成资源后，您有责任删除日志传输资源。为此，请按照以下步骤操作。

1. 使用 [DeleteDelivery](#) 操作删除 Delivery。
2. 使用 [DeleteDeliverySource](#) 操作删除 DeliverySource。
3. 如果与您刚刚删除的 DeliverySource 关联的 DeliveryDestination 仅用于此特定的 DeliverySource，则您可以使用 [DeleteDeliveryDestinations](#) 操作将其删除。

目录

- [发送到 CloudWatch Logs 的日志](#)
- [发送到 Amazon S3 的日志](#)
 - [Amazon S3 存储桶服务器端加密](#)
- [日志发送至 Firehose](#)
- [跟踪发送到 X-Ray](#)

发送到 CloudWatch Logs 的日志

用户权限

要启用将日志发送到 CloudWatch Logs，您必须使用以下权限登录。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadWriteAccessForLogDeliveryActions",
      "Effect": "Allow",
      "Action": [
        "logs:GetDelivery",
        "logs:GetDeliverySource",
        "logs:PutDeliveryDestination",
        "logs:GetDeliveryDestinationPolicy",
        "logs:DeleteDeliverySource",
        "logs:PutDeliveryDestinationPolicy",
```

```

        "logs:CreateDelivery",
        "logs:GetDeliveryDestination",
        "logs:PutDeliverySource",
        "logs>DeleteDeliveryDestination",
        "logs>DeleteDeliveryDestinationPolicy",
        "logs>DeleteDelivery",
        "logs:UpdateDeliveryConfiguration"
    ],
    "Resource": [
        "arn:aws:logs:us-east-1:111122223333:delivery:*",
        "arn:aws:logs:us-east-1:444455556666:delivery-source:*",
        "arn:aws:logs:us-east-1:777788889999:delivery-destination:*"
    ]
},
{
    "Sid": "ListAccessForLogDeliveryActions",
    "Effect": "Allow",
    "Action": [
        "logs:DescribeDeliveryDestinations",
        "logs:DescribeDeliverySources",
        "logs:DescribeDeliveries",
        "logs:DescribeConfigurationTemplates"
    ],
    "Resource": "*"
},
{
    "Sid": "AllowUpdatesToResourcePolicyCWL",
    "Effect": "Allow",
    "Action": [
        "logs:PutResourcePolicy",
        "logs:DescribeResourcePolicies",
        "logs:DescribeLogGroups"
    ],
    "Resource": [
        "arn:aws:logs:us-east-1:123456789012:*"
    ]
}
]
}

```

日志组和资源策略

接收日志的日志组必须具有包含特定权限的资源策略。如果日志组当前没有资源策略，而且设置日志记录的用户对日志组具有 `logs:PutResourcePolicy`、`logs:DescribeResourcePolicies` 和 `logs:DescribeLogGroups` 权限，那么当您开始将日志发送到 CloudWatch Logs 日志时，Amazon 会自动为其创建以下策略。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite20150319",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "delivery.logs.amazonaws.com"
        ]
      },
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group:log-  
stream:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": [
            "0123456789"
          ]
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:logs:us-east-1:111122223333:*"
          ]
        }
      }
    }
  ]
}
```

发送到 Amazon S3 的日志

用户权限

要启用向 Amazon S3 发送日志，您必须使用以下权限登录。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadWriteAccessForLogDeliveryActions",
      "Effect": "Allow",
      "Action": [
        "logs:GetDelivery",
        "logs:GetDeliverySource",
        "logs:PutDeliveryDestination",
        "logs:GetDeliveryDestinationPolicy",
        "logs>DeleteDeliverySource",
        "logs:PutDeliveryDestinationPolicy",
        "logs>CreateDelivery",
        "logs:GetDeliveryDestination",
        "logs:PutDeliverySource",
        "logs>DeleteDeliveryDestination",
        "logs>DeleteDeliveryDestinationPolicy",
        "logs>DeleteDelivery",
        "logs:UpdateDeliveryConfiguration"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:111122223333:delivery:*",
        "arn:aws:logs:us-east-1:111122223333:delivery-source:*",
        "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
      ]
    },
    {
      "Sid": "ListAccessForLogDeliveryActions",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeDeliveryDestinations",
        "logs:DescribeDeliverySources",
        "logs:DescribeDeliveries",
        "logs:DescribeConfigurationTemplates"
      ]
    }
  ]
}
```

```

    ],
    "Resource": "*"
  },
  {
    "Sid": "AllowUpdatesToResourcePolicyS3",
    "Effect": "Allow",
    "Action": [
      "s3:PutBucketPolicy",
      "s3:GetBucketPolicy"
    ],
    "Resource": "arn:aws:s3:::bucket-name"
  }
]
}

```

接收日志的 S3 存储桶必须具有包含特定权限的资源策略。如果存储桶当前没有资源策略，而且设置日志记录的用户对存储桶具有 S3:GetBucketPolicy 和 S3:PutBucketPolicy 权限，那么当您开始将日志发送到 Amazon S3 时，Amazon 会自动为其创建以下策略。

JSON

```

{
  "Version": "2012-10-17",
  "Id": "AWSLogDeliveryWrite20150319",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/AWSLogs/account-ID/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control",
          "aws:SourceAccount": [
            "0123456789"
          ]
        }
      },
      "ArnLike": {

```

```
        "aws:SourceArn": [
            "arn:aws:logs:us-east-1:111122223333:delivery-source:*"
        ]
    }
}
```

在之前的策略中，对于 `aws:SourceAccount`，请指定将日志传送到此存储桶的账户 ID 列表。对于 `aws:SourceArn`，请按 `arn:aws:logs:source-region:source-account-id:*` 格式指定生成日志的资源 ARN 列表。

如果存储桶具有资源策略，但该策略不包含上一个策略中所示的语句，并且设置日志记录的用户对存储桶具有 `S3:GetBucketPolicy` 和 `S3:PutBucketPolicy` 权限，则该语句将附加到存储桶的资源策略中。

Note

某些情况下，如果未向 `delivery.logs.amazonaws.com` 授予 `s3:ListBucket` 权限，可能会在 Amazon CloudTrail 中看到 `AccessDenied` 错误。为了避免 CloudTrail 日志中出现此类错误，必须向 `delivery.logs.amazonaws.com` 授予 `s3:ListBucket` 权限且必须包含与在上述存储桶策略中设置的 `s3:GetBucketAcl` 权限一起显示的 `Condition` 参数。为方便起见，可以直接将 `AWSLogDeliveryAclCheck` 更新为 `"Action": ["s3:GetBucketAcl", "s3:ListBucket"]`，而不是创建一个新的 `Statement`

Amazon S3 存储桶服务器端加密

您可以通过启用 Amazon S3 托管式密钥 (SSE-S3) 的服务器端加密或 Amazon Key Management Service 中存储 Amazon KMS 密钥 (SSE-KMS) 的服务器端加密来保护 Amazon S3 存储桶中的数据。有关更多信息，请参阅[使用服务器端加密保护数据](#)。

如果选择 SSE-S3，则不需要额外的配置。Amazon S3 处理加密密钥。

Warning

如果选择 SSE-KMS，则必须使用客户托管式密钥，因为此场景不支持使用 Amazon 托管式密钥。如果您使用 Amazon 托管密钥设置加密，则会以不可读取的格式提供日志。

当您使用客户托管式 Amazon KMS 密钥时，您可以在启用存储桶加密时指定客户托管式密钥的 Amazon Resource Name (ARN)。您必须将以下内容添加到客户托管式密钥的密钥策略（不是 S3 存储桶的存储桶策略）中，以便日志传输账户可以写入 S3 存储桶。

如果选择 SSE-KMS，则必须使用客户托管式密钥，因为此场景不支持使用 Amazon 托管式密钥。当您使用客户托管式 Amazon KMS 密钥时，您可以在启用存储桶加密时指定客户托管式密钥的 Amazon Resource Name (ARN)。您必须将以下内容添加到客户托管式密钥的密钥策略（不是 S3 存储桶的存储桶策略）中，以便日志传输账户可以写入 S3 存储桶。

```
{
  "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:*"]
    }
  }
}
```

对于 `aws:SourceAccount`，请指定将日志传送到此存储桶的账户 ID 列表。对于 `aws:SourceArn`，请按 `arn:aws:logs:source-region:source-account-id:*` 格式指定生成日志的资源 ARN 列表。

日志发送至 Firehose

用户权限

要启用向 Firehose 发送日志，您必须使用以下权限登录。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadWriteAccessForLogDeliveryActions",
      "Effect": "Allow",
      "Action": [
        "logs:GetDelivery",
        "logs:GetDeliverySource",
        "logs:PutDeliveryDestination",
        "logs:GetDeliveryDestinationPolicy",
        "logs>DeleteDeliverySource",
        "logs:PutDeliveryDestinationPolicy",
        "logs>CreateDelivery",
        "logs:GetDeliveryDestination",
        "logs:PutDeliverySource",
        "logs>DeleteDeliveryDestination",
        "logs>DeleteDeliveryDestinationPolicy",
        "logs>DeleteDelivery",
        "logs:UpdateDeliveryConfiguration"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:111122223333:delivery:*",
        "arn:aws:logs:us-east-1:111122223333:delivery-source:*",
        "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
      ]
    },
    {
      "Sid": "ListAccessForLogDeliveryActions",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeDeliveryDestinations",
        "logs:DescribeDeliverySources",
        "logs:DescribeDeliveries",
        "logs:DescribeConfigurationTemplates"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowUpdatesToResourcePolicyFH",
      "Effect": "Allow",

```



```

        "Action": [
            "firehose:TagDeliveryStream"
        ],
        "Resource": [
            "arn:aws:firehose:us-east-1:111122223333:deliverystream/*"
        ]
    },
    {
        "Sid": "CreateServiceLinkedRole",
        "Effect": "Allow",
        "Action": [
            "iam:CreateServiceLinkedRole"
        ],
        "Resource": "arn:aws:iam::111122223333:role/aws-service-role/delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery"
    }
]
}

```

用于资源权限的 IAM 角色

由于 Firehose 不使用资源策略，Amazon 将这些日志设置为发送到 Firehose 时会使用 IAM 角色。Amazon 创建名为 `AWSServiceRoleForLogDelivery` 的服务相关角色。此服务相关角色包括以下权限。

JSON

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch",
                "firehose:ListTagsForDeliveryStream"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/LogDeliveryEnabled": "true"
                }
            }
        }
    ]
}

```

```
    },
    "Effect": "Allow"
  }
]
```

此服务相关角色会为将 LogDeliveryEnabled 标签设置为 true 的所有 Firehose 传输流授予权限。当您设置日志记录时，Amazon 会将此标签提供给目标传输流。

此服务相关角色还具有允许 delivery.logs.amazonaws.com 服务委托人来代入所需服务相关角色的信任策略。该信任策略如下所示：

JSON

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

跟踪发送到 X-Ray

用户权限

要将跟踪发送到 Amazon X-Ray，您必须使用以下权限登录。

JSON

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

```

        "Sid": "ReadWriteAccessForLogDeliveryActions",
        "Effect": "Allow",
        "Action": [
            "logs:GetDelivery",
            "logs:GetDeliverySource",
            "logs:PutDeliveryDestination",
            "logs:GetDeliveryDestinationPolicy",
            "logs>DeleteDeliverySource",
            "logs:PutDeliveryDestinationPolicy",
            "logs:CreateDelivery",
            "logs:GetDeliveryDestination",
            "logs:PutDeliverySource",
            "logs>DeleteDeliveryDestination",
            "logs>DeleteDeliveryDestinationPolicy",
            "logs>DeleteDelivery",
            "logs:UpdateDeliveryConfiguration"
        ],
        "Resource": [
            "arn:aws:logs:us-east-1:111122223333:delivery:*",
            "arn:aws:logs:us-east-1:111122223333:delivery-source:*",
            "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
        ]
    },
    {
        "Sid": "ListAccessForLogDeliveryActions",
        "Effect": "Allow",
        "Action": [
            "logs:DescribeDeliveryDestinations",
            "logs:DescribeDeliverySources",
            "logs:DescribeDeliveries",
            "logs:DescribeConfigurationTemplates"
        ],
        "Resource": "*"
    },
    {
        "Sid": "AllowUpdatesToResourcePolicyXRay",
        "Effect": "Allow",
        "Action": [
            "xray:PutResourcePolicy",
            "xray:ListResourcePolicies"
        ],
        "Resource": "*"
    }
]

```

```
}
```

X-Ray 资源策略

接收跟踪的目标账户必须具有包含特定权限的资源策略。当 X-Ray 当前没有资源策略，并且设置跟踪的用户在账户中拥有 `xray:PutResourcePolicy` 和 `xray:ListResourcePolicies` 权限时，Amazon 将在您开始向 X-Ray 发送跟踪时自动创建以下策略。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "xray:PutTraceSegments",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ForAllValues:ArnLike": {
          "logs:LogGeneratingResourceArns": "arn:aws:bedrock-agentcore:us-east-1:123456789012:memory/MemoryId"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:logs:us-east-1:123456789012:delivery-source:xray-test"
        }
      }
    }
  ]
}
```

启用事务搜索

要将跟踪发送到 X-Ray，您必须启用[事务搜索](#)。

服务特定的权限

除了前面部分列出的特定于目标的权限之外，一些服务还需要明确授权，以允许客户从其资源发送日志，以此作为额外安全层。它授权在该服务中公开发布日志的资源执行 `AllowVendedLogDeliveryForResource` 操作。对于这些服务，请使用以下策略并将 *service* 和 *resource-type* 替换为适当的值。有关这些字段的服务特定值，请参阅这些服务的公开发布的日志的文档页面。在以下示例中，策略已更新为启用来自 Amazon SES 的公开发布的日志。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ServiceLevelAccessForLogDelivery",
      "Effect": "Allow",
      "Action": [
        "ses:AllowVendedLogDeliveryForResource"
      ],
      "Resource": "arn:aws:ses:us-east-1:123456789012:resource-type/*"
    }
  ]
}
```

控制台特定的权限

除了前面部分列出的权限外，如果您使用控制台而不是 API 设置日志传输，则还需要以下额外权限：

JSON

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

```

        "logs:DescribeLogGroups",
        "logs:CreateLogGroup"
    ],
    "Resource": [
        "arn:aws:logs:us-east-1:111122223333:log-group:*"
    ]
},
{
    "Sid": "AllowLogDeliveryActionsConsoleS3",
    "Effect": "Allow",
    "Action": [
        "s3:ListAllMyBuckets",
        "s3:ListBucket",
        "s3:GetBucketLocation"
    ],
    "Resource": [
        "arn:aws:s3:::*"
    ]
},
{
    "Sid": "AllowLogDeliveryActionsConsoleFH",
    "Effect": "Allow",
    "Action": [
        "firehose:ListDeliveryStreams",
        "firehose:DescribeDeliveryStream"
    ],
    "Resource": [
        "*"
    ]
}
]
}

```

跨账户传输示例

在此示例中，涉及两个账户。具有日志生成资源的账户是账户 A，ID：**123456789012**，具有日志消耗资源的账户是账户 B，ID：**111122223333**。

账户 A 想要使用 ARN `arn:aws:bedrock:us-east-1:123456789012:knowledge-base/kb-12345678` 从其账户中的 Amazon Bedrock 知识库传输日志。

对于此示例，账户 A 需要以下权限：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowVendedLogDeliveryForKnowledgeBase",
      "Effect": "Allow",
      "Action": [
        "bedrock:AllowVendedLogDeliveryForResource"
      ],
      "Resource": "arn:aws:bedrock:us-east-1:123456789012:knowledge-
base/XXXXXXXXXX"
    },
    {
      "Sid": "CreateLogDeliveryPermissions",
      "Effect": "Allow",
      "Action": [
        "logs:PutDeliverySource",
        "logs:CreateDelivery"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:delivery-source:*",
        "arn:aws:logs:us-east-1:123456789012:delivery:*",
        "arn:aws:logs:us-east-1:444455556666:delivery-destination:*"
      ]
    }
  ]
}
```

创建传输源

首先，账户 A 使用其 bedrock 知识库创建传输源：

```
aws logs put-delivery-source --name my-delivery-source --log-type APPLICATION_LOGS --
resource-arn arn:aws:bedrock:region:AAAAAAAAAAAA:knowledge-base/XXXXXXXXXX
```

接下来，账户 B 必须使用以下流之一创建传输目标：

- [配置传输到 Amazon S3 存储桶](#)

- [配置向 Firehose 流的传输](#)

配置传输到 Amazon S3 存储桶

用户 B 希望使用 ARN `arn:aws:s3:::amzn-s3-demo-bucket` 将日志接收到其 S3 存储桶中。对于此示例，账户 B 将需要以下权限：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PutLogDestinationPermissions",
      "Effect": "Allow",
      "Action": [
        "logs:PutDeliveryDestination",
        "logs:PutDeliveryDestinationPolicy"
      ],
      "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
    }
  ]
}
```

存储桶在其存储桶策略中需要具有以下权限：

JSON

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



```

        "s3:PutObject"
    ],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/AWSLogs/123456789012/
*",
    "Condition": {
        "StringEquals": {
            "s3:x-amz-acl": "bucket-owner-full-control",
            "aws:SourceAccount": [
                "123456789012"
            ]
        },
        "ArnLike": {
            "aws:SourceArn": [
                "arn:aws:logs:us-east-1:123456789012:delivery-source:my-
delivery-source"
            ]
        }
    }
}
]
}

```

如果存储桶使用 SSE-KMS 进行加密，请确保 Amazon KMS 密钥政策具有相应的权限。例如，如果 KMS 密钥是 `arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`，请使用以下内容：

JSON

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowLogsGenerateDataKey",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey"
            ],

```

```

    "Resource": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": [
          "123456789012"
        ]
      },
      "ArnLike": {
        "aws:SourceArn": [
          "arn:aws:logs:us-east-1:123456789012:delivery-source:my-
delivery-source"
        ]
      }
    }
  }
]
}

```

然后，账户 B 可以创建一个以 S3 存储桶为目标资源的传输目标：

```
aws logs put-delivery-destination --name my-s3-delivery-destination --delivery-
destination-configuration "destinationResourceArn=arn:aws:s3:::amzn-s3-demo-bucket"
```

接下来，账户 B 在其新创建的传输目标上创建传输目标策略，该策略将授予账户 A 创建日志传输的权限。将添加到新创建的传输目标的策略如下：

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateDelivery",
      "Effect": "Allow",
      "Principal": {
        "AWS": "123456789012"
      },
      "Action": [
        "logs:CreateDelivery"
      ]
    }
  ]
}

```

```

    ],
    "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:amzn-s3-demo-bucket"
  }
]
}

```

此策略将以 `destination-policy-s3.json` 形式保存在账户 B 的计算机中。要附加此资源，账户 B 将运行以下命令：

```
aws logs put-delivery-destination-policy --delivery-destination-name my-s3-delivery-
destination --delivery-destination-policy file://destination-policy-s3.json
```

最后，账户 A 创建传输，将账户 A 中的传输源链接到账户 B 中的传输目标。

```
aws logs create-delivery --delivery-source-name my-delivery-source --delivery-
destination-arn arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:my-s3-delivery-
destination
```

配置向 Firehose 流的传输

在此示例中，账户 B 希望将日志接收到其 Firehose 流中。Firehose 流具有以下 ARN，并配置为使用 DirectPut 传输流类型：

`arn:aws:firehose:us-east-1:111122223333:deliverystream/log-delivery-stream`

对于此示例，账户 B 需要以下权限：

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFirehoseCreateSLR",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],

```

```

        "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery"
    },
    {
        "Sid": "AllowFirehoseTagging",
        "Effect": "Allow",
        "Action": [
            "firehose:TagDeliveryStream"
        ],
        "Resource": "arn:aws:firehose:us-
east-1:111122223333:deliverystream/X"
    },
    {
        "Sid": "AllowFirehoseDeliveryDestination",
        "Effect": "Allow",
        "Action": [
            "logs:PutDeliveryDestination",
            "logs:PutDeliveryDestinationPolicy"
        ],
        "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:*"
    }
]
}

```

Firehose 流必须将标签 LogDeliveryEnabled 设置为 true。

然后，账户 B 将创建一个以 Firehose 流为目标资源的传输目标：

```

aws logs put-delivery-destination --name my-fh-delivery-destination --delivery-
destination-configuration
    "destinationResourceArn=arn:aws:firehose:region:BBBBBBBBBBBB:deliverystream/X"

```

接下来，账户 B 在其新创建的传输目标上创建传输目标策略，该策略将授予账户 A 创建日志传输的权限。要添加到新创建的传输目标的策略如下：

JSON

```

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

```

```
{
  "Sid": "AllowCreateDelivery",
  "Effect": "Allow",
  "Principal": {
    "AWS": "123456789012"
  },
  "Action": [
    "logs:CreateDelivery"
  ],
  "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-destination:amzn-s3-demo-bucket"
}
```

此策略将以 `destination-policy-fh.json` 形式保存在账户 B 的计算机中。要附加此资源，账户 B 运行以下命令：

```
aws logs put-delivery-destination-policy --delivery-destination-name my-fh-delivery-destination --delivery-destination-policy file://destination-policy-fh.json
```

最后，账户 A 创建传输，将账户 A 中的传输源链接到账户 B 中的传输目标。

```
aws logs create-delivery --delivery-source-name my-delivery-source --delivery-destination-arn arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:my-fh-delivery-destination
```

防止跨服务混淆座席

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在 Amazon 中，跨服务模拟可能会导致混淆代理问题。一个服务（呼叫服务）调用另一项服务（所谓的 *服务*）时，可能会发生跨服务模拟。可以操纵调用服务，使用其权限以在其他情况下该服务不应有访问权限的方式对另一个客户的资源进行操作。为防止这种情况，Amazon 提供可帮助您保护所有服务的数据的工具，而这些服务中的服务主体有权限访问账户中的资源。

我们建议在资源策略中使用 [aws:SourceArn](#)、[aws:SourceAccount](#)、[aws:SourceOrgID](#) 和 [aws:SourceOrgPaths](#) 全局条件上下文键，以限制 CloudWatch Logs 为其他服务提供的资源访问权限。使用 `aws:SourceArn` 来仅将一个资源与跨服务访问相关联。使用 `aws:SourceAccount` 来让该账户中的任何资源与跨服务使用相关联。使用 `aws:SourceOrgID` 来允许某组织内的任何账户

中的任何资源与跨服务使用相关联。使用 `aws:SourceOrgPaths` 来将 Amazon Organizations 路径内账户中的任何资源与跨服务使用相关联。有关使用和了解路径的更多信息，请参阅[了解 Amazon Organizations 实体路径](#)。

防范混淆代理问题最有效的方法是使用 `aws:SourceArn` 全局条件上下文键和资源的完整 ARN。如果不知道资源的完整 ARN，或者正在指定多个资源，请针对 ARN 未知部分使用带有通配符字符 (*) 的 `aws:SourceArn` 全局上下文条件键。例如 `arn:aws:servicename:*:123456789012:*`。

如果 `aws:SourceArn` 值不包含账户 ID，例如 Amazon S3 桶 ARN，您必须使用 `aws:SourceAccount` 和 `aws:SourceArn` 来限制权限。

要防范大规模混淆代理问题，请在基于资源的策略中将 `aws:SourceOrgID` 或 `aws:SourceOrgPaths` 全局条件上下文键与资源的组织 ID 或组织路径一起使用。包含 `aws:SourceOrgID` 或 `aws:SourceOrgPaths` 键的策略将自动包含正确的账户，并且当您在组织中添加、删除或移动账户时，无需手动更新策略。

[步骤 1：创建目标](#)和[步骤 2：创建目标](#)中所述的用于向 CloudWatch Logs 授予将数据写入 Kinesis Data Streams 和 Firehose 的权限的策略展示了如何使用 `aws:SourceArn` 全局条件上下文键来帮助防止混淆代理问题。

将日志数据导出到 Amazon S3

本章为您提供了信息，以便您可以将日志数据从日志组导出到 Amazon S3 存储桶进行自定义处理和分析，或加载到其他系统。您可以将数据导出到同一账户或其他账户中的存储桶。

您可执行以下操作：

- 将日志数据导出到由 Amazon Key Management Service (Amazon KMS) 中的 SSE-KMS 加密的 S3 存储桶
- 将日志数据导出到已启用 S3 对象锁定且具有保留期的 S3 桶

Note

仅标准日志类中的日志组支持导出到 Amazon S3。有关日志类的更多信息，请参阅 [日志类](#)。

我们建议您不要定期将日志导出到 Amazon S3 以持续归档日志。对于该使用案例，我们建议您使用订阅。有关订阅的更多信息，请参阅[使用订阅实时处理日志数据](#)。

要开始导出进程，您必须创建一个用于存储导出的日志数据的 S3 存储桶。您可以将已导出的文件存储在您的 S3 桶中，并定义 Amazon S3 生命周期规则以自动存档或删除已导出的文件。

您可以导出到使用 AES-256 或 SSE-KMS 加密的 S3 存储桶。不支持导出到由 DSSE-KMS 加密的存储桶。

可将多个日志组或多个时间范围的日志导出至同一个 S3 存储桶中。要分开每个导出任务的日志数据，您可以指定一个前缀，以使用作所有导出对象的 Amazon S3 键前缀。

Note

不确保会对导出文件中的日志数据块进行基于时间的排序。您可以使用 Linux 实用程序对导出的日志字段数据进行排序。例如，以下实用程序命令对单个文件夹中所有 .gz 文件中的事件进行排序。

```
find . -exec zcat {} + | sed -r 's/^[0-9]+\x00&/' | sort -z
```

以下实用程序命令对多个子文件夹中的 .gz 文件进行排序。

```
find ./ */ -type f -exec zcat {} + | sed -r 's/^[0-9]+\x0&/' | sort -z
```

此外，您还可以使用其他 `stdout` 命令将排序后的输出通过管道传输到另一个文件进行保存。

日志数据可能需要长达 12 小时才能用于导出。导出任务在 24 小时后超时。如果导出任务超时，请在创建导出任务时缩短时间范围。

有关日志数据的近实时分析，请参阅 [使用 CloudWatch Logs Insights 分析日志数据](#) 或 [使用订阅实时处理日志数据](#)。

目录

- [概念](#)
- [使用控制台将日志数据导出到 Amazon S3](#)
- [使用 Amazon CLI 将日志数据导出到 Amazon S3](#)
- [描述导出任务](#)
- [取消导出任务](#)

概念

开始之前，请熟悉以下导出概念：

日志组名称

与导出任务关联的日志组的名称。该日志组中的日志数据将导出至指定的 S3 桶中。

从 (时间戳)

必填的时间戳，表示自 1970 年 1 月 1 日 00:00:00 UTC 以来的毫秒数。日志组中在此时间或之后提取的所有日志事件都将被导出。

至 (时间戳)

必填的时间戳，表示自 1970 年 1 月 1 日 00:00:00 UTC 以来的毫秒数。日志组中该时间之前获得的所有日志事件都会被导出。

目标存储桶

与导出任务关联的 S3 桶的名称。此存储桶用于导出指定日志组中的日志数据。

目标前缀

可选属性，用作所有导出对象的 Amazon S3 键前缀。这有助于在您的存储桶中创建类似于文件夹的组织结构。

使用控制台将日志数据导出到 Amazon S3

在以下示例中，您将使用 Amazon CloudWatch 控制台将所有数据从名为 my-log-group 的 Amazon CloudWatch Logs 日志组导出到名为 my-exported-logs 的 Amazon S3 存储桶。

支持将日志数据导出到由 SSE-KMS 加密的 S3 存储桶。不支持导出到由 DSSE-KMS 加密的存储桶。

有关如何设置导出的详细信息取决于您要导出到的 Amazon S3 存储桶与要导出的日志是位于同一个账户中，还是不同的账户中。

主题

- [同账号导出](#)
- [跨账户导出](#)

同账号导出

如果 Amazon S3 存储桶与要导出的日志位于同一个账户中，请参阅本节中的说明。

主题

- [步骤 1：创建 Amazon S3 存储桶](#)
- [步骤 2：设置访问权限](#)
- [步骤 3：设置 Amazon S3 存储桶上的权限](#)
- [\(可选 \) 步骤 4：导出到使用 SSE-KMS 加密的桶](#)
- [步骤 5：创建导出任务](#)

步骤 1：创建 Amazon S3 存储桶

我们建议您使用专为 CloudWatch Logs 创建的存储桶。但是，如果要使用现有存储桶，请跳至第 2 步。

 Note

Amazon S3 存储桶必须与要导出的日志数据位于同一区域。CloudWatch Logs 不支持将数据导出到其他区域中的 Amazon S3 存储桶。

创建 Amazon S3 存储桶

1. 通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 如果需要，可以更改区域。从导航栏中，选择 CloudWatch Logs 所在的区域。
3. 选择 Create Bucket (创建存储桶)。
4. 对于 Bucket Name (存储桶名称)，输入存储桶的名称。
5. 对于 Region (区域)，选择 CloudWatch Logs 数据所在的区域。
6. 选择创建。

步骤 2：设置访问权限

要在步骤 5 中创建导出任务，您需要分配 AmazonS3ReadOnlyAccess IAM 角色和以下权限：

- logs:CreateExportTask
- logs:CancelExportTask
- logs:DescribeExportTasks
- logs:DescribeLogStreams
- logs:DescribeLogGroups

要提供访问权限，请为您的用户、组或角色添加权限：

- 通过身份提供商在 IAM 中托管的用户：

创建适用于身份联合验证的角色。按照《IAM 用户指南》中[针对第三方身份提供商创建角色 \(联合身份验证 \)](#)的说明进行操作。

- IAM 用户：

- 创建您的用户可以担任的角色。按照《IAM 用户指南》中[为 IAM 用户创建角色](#)的说明进行操作。
- (不推荐使用) 将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中[向用户添加权限 \(控制台 \)](#)中的说明进行操作。

步骤 3：设置 Amazon S3 存储桶上的权限

默认情况下，所有 Amazon S3 存储桶和对象都是私有的。仅资源所有者（创建了桶的 Amazon Web Services 账户）能够访问桶及其包含的任何对象。不过，资源所有者可以选择通过编写访问策略来向其他资源和用户授予访问权限。

当您设置策略时，建议包含一个随机生成的字符串作为存储桶的前缀，这样只会将目标日志流导出到该存储桶。

Important

为了使导出到 Amazon S3 存储桶更加安全，我们现在要求您指定允许将日志数据导出到 S3 桶的源账户列表。

在以下示例中，aws:SourceAccount 密钥中账户 ID 的列表是用户可以从其中将日志数据导出到 Amazon S3 存储桶的账户。aws:SourceArn 密钥是正在进行的操作的资源。如本示例所示，您可以将其限制为特定的日志组，也可以使用通配符。

建议您还包括创建 S3 桶的账户的账户 ID，以允许在同一账户内导出。

设置 Amazon S3 存储桶上的权限

1. 在 Amazon S3 控制台中，选择您在第 1 步中创建的存储桶。
2. 选择 Permissions（权限）、Bucket policy（存储桶策略）。
3. 在 Bucket Policy Editor（桶策略编辑器）中，添加以下策略。将 my-exported-logs 更改为 S3 存储桶的名称。请务必指定正确的区域端点，例如 Principal（委托人）的 us-west-1。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCloudWatchLogsGetBucketAcl",
      "Action": "s3:GetBucketAcl",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
      "Condition": {
```

```

        "StringEquals": {
            "aws:SourceAccount": [
                "123456789012",
                "111122223333"
            ]
        },
        "ArnLike": {
            "aws:SourceArn": [
                "arn:aws:logs:us-east-1:123456789012:log-group:*",
                "arn:aws:logs:us-east-1:111122223333:log-group:*"
            ]
        }
    },
    {
        "Sid": "AllowCloudWatchLogsPutObject",
        "Action": "s3:PutObject",
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
        "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control",
                "aws:SourceAccount": [
                    "123456789012",
                    "111122223333"
                ]
            },
            "ArnLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:us-east-1:123456789012:log-group:*",
                    "arn:aws:logs:us-east-1:111122223333:log-group:*"
                ]
            }
        }
    }
]
}

```

4. 选择 Save，将您刚添加的策略设置为存储桶上的访问策略。此策略可让 CloudWatch Logs 将日志数据导出到您的 Amazon S3 存储桶中。存储桶所有者对所有导出的对象拥有完全权限。

 Warning

如果现有存储桶已附加一个或多个策略，请将 CloudWatch Logs 访问的语句添加到这些策略。我们建议您评估生成的权限集，确保它们适合访问存储桶的用户。

(可选) 步骤 4：导出到使用 SSE-KMS 加密的桶

只有当您要导出到使用 Amazon KMS keys 服务器端加密的 Amazon S3 存储桶时，才需要执行此步骤。这种加密称为 SSE-KMS。

导出到使用 SSE-KMS 加密的桶

1. 从 <https://console.aws.amazon.com/kms> 打开 Amazon KMS 控制台。
2. 要更改 Amazon Web Services 区域，请使用页面右上角的区域选择器。
3. 在左侧导航栏中，选择 Customer managed keys (客户托管密钥)。

选择 Create Key (创建密钥)。

4. 对于 Key type (密钥类型)，选择 Symmetric (对称)。
5. 对于 Key usage (密钥用法)，选择 Encrypt and decrypt (加密和解密)，然后选择 Next (下一步)。
6. 在 Add labels (添加标签) 下，输入密钥的别名，并添加描述或标签 (可选)。然后选择下一步。
7. 在 Key administrators (密钥管理员) 下，选择可以管理此密钥的人员，然后选择 Next (下一步)。
8. 在 Define key usage permissions (定义密钥使用权限) 下，不进行任何更改并选择 Next (下一步)。
9. 检查设置，然后选择 Finish (完成)。
10. 返回 Customer managed keys (客户托管密钥) 页面，选择您刚刚创建的密钥的名称。
11. 选择 Key policy (密钥策略) 选项卡，然后选择 Switch to policy view (切换到策略视图)。
12. 在 Key policy (密钥策略) 部分，选择 Edit (编辑)。
13. 将以下语句添加到密钥策略语句列表。执行此操作时，将 *Region* 替换为日志的区域，将 *account-ARN* 替换为拥有 KMS 密钥的账户的 ARN。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow CWL Service Principal usage",
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.Region.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "account-ARN"
      },
      "Action": [
        "kms:GetKeyPolicy*",
        "kms:PutKeyPolicy*",
        "kms:DescribeKey*",
        "kms:CreateAlias*",
        "kms:ScheduleKeyDeletion*",
        "kms:Decrypt"
      ],
      "Resource": "*"
    }
  ]
}
```

14. 选择保存更改。
15. 通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
16. 找到您在 [步骤 1：创建 S3 存储桶](#) 中创建的桶并选择桶名称。
17. 选择属性选项卡。然后，在 Default Encryption (默认加密) 下，选择 Edit (编辑)。

18. 在 Server-side Encryption (服务器端加密) 下 , 选择 Enable (启用) 。
19. 在 Encryption type (加密类型) 下 , 请选择 Amazon Key Management Service key (SSE-KMS) (Amazon Key Management Service 密钥(SSE-KMS))。
20. 选择 Choose from your Amazon KMS keys (从密钥中选择) , 然后找到您创建的密钥。
21. 对于 Bucket Key (桶密钥) , 选择 Enable (启用) 。
22. 选择保存更改。

步骤 5 : 创建导出任务

在本步骤中 , 您可创建导出任务以便从日志组中导出日志。

使用 CloudWatch 控制台将数据导出到 Amazon S3

1. 如 [步骤 2 : 设置访问权限](#) 中所述 , 使用足够的权限登录。
2. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
3. 在导航窗格中 , 选择 Log groups (日志组) 。
4. 在 Log Groups (日志组) 屏幕上 , 选择日志组的名称。
5. 选择 Actions (操作) 、 Export data to Amazon S3 (将数据导出到 Amazon S3) 。
6. 在 Export data to Amazon S3 (将数据导出到 Amazon S3) 屏幕的 Define data export (定义数据导出) 下 , 使用 From (从) 和 To (至) 设置要导出的数据的时间范围。
7. 如果日志组有多个日志流 , 您可以提供一个日志流前缀 , 将日志组数据限定为特定流。选择 Advanced (高级) , 然后为 Stream prefix (流前缀)输入日志流前缀。
8. 在 Choose S3 bucket (选择 S3 桶) 下 , 选择与 S3 桶关联的账户。
9. 对于 S3 存储桶名称 , 请选择一个 S3 存储桶。
10. 对于 S3 Bucket prefix (S3 存储桶前缀) , 输入在存储桶策略中指定的随机生成的字符串。
11. 选择 Export (导出) , 将日志数据导出到 Amazon S3。
12. 要查看您导出到 Amazon S3 的日志数据的状态 , 请依次选择 Actions (操作) 、 View all exports to Amazon S3 (查看导出到 Amazon S3 的所有内容) 。

跨账户导出

如果 Amazon S3 存储桶与要导出的日志位于不同账户中 , 请参阅本节中的说明。

主题

- [步骤 1：创建 Amazon S3 存储桶](#)
- [步骤 2：设置访问权限](#)
- [步骤 3：在 S3 桶上设置权限](#)
- [\(可选 \) 步骤 4：导出到使用 SSE-KMS 加密的桶](#)
- [步骤 5：创建导出任务](#)

步骤 1：创建 Amazon S3 存储桶

我们建议您使用专为 CloudWatch Logs 创建的存储桶。但是，如果要使用现有存储桶，请跳至第 2 步。

Note

Amazon S3 存储桶必须与要导出的日志数据位于同一区域。CloudWatch Logs 不支持将数据导出到其他区域中的 Amazon S3 存储桶。

创建 Amazon S3 存储桶

1. 通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 如果需要，可以更改区域。从导航栏中，选择 CloudWatch Logs 所在的区域。
3. 选择 Create Bucket (创建存储桶)。
4. 对于 Bucket Name (存储桶名称)，输入存储桶的名称。
5. 对于 Region (区域)，选择 CloudWatch Logs 数据所在的区域。
6. 选择创建。

步骤 2：设置访问权限

首先，您必须创建新的 IAM 策略，以使 CloudWatch Logs 对目标账户中的目标 Amazon S3 存储桶执行 s3:PutObject 操作。

策略中包含的其他操作取决于目标存储桶是否使用 Amazon KMS 加密，或者是否使用 [S3 对象所有权](#) 设置启用了 ACL。

- 如果使用 KMS 加密，请为密钥资源添加 GenerateDataKey 和 Decrypt 操作
- 如果在存储桶上启用了 ACL，请为存储桶资源添加 s3:PutObjectAcl 操作

创建 IAM policy 以将日志导出到 Amazon S3 存储桶

1. 通过 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在左侧的导航窗格中，选择策略。
3. 选择创建策略。
4. 在策略编辑器部分，选择 JSON。
5. 如果目标存储桶不使用 Amazon KMS 加密，请将以下策略粘贴到编辑器中。

JSON

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

如果目标存储桶使用 Amazon KMS 加密，请将以下策略粘贴到编辑器中。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ]
    }
  ]
}
```

```
"Resource": "arn:aws:kms:us-east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ]
}
```

6. 选择下一步。
7. 输入策略名称。您将使用此名称将策略附加到您的 IAM 角色。
8. 选择创建策略以保存新策略。

要在步骤 5 中创建导出任务，您需要使用 AmazonS3ReadOnlyAccess IAM 角色登录。您还必须使用刚刚创建的 IAM policy 登录并且具有以下权限：

- logs:CreateExportTask
- logs:CancelExportTask
- logs:DescribeExportTasks
- logs:DescribeLogStreams
- logs:DescribeLogGroups

要提供访问权限，请为您的用户、组或角色添加权限：

- 通过身份提供商在 IAM 中托管的用户：

创建适用于身份联合验证的角色。按照《IAM 用户指南》中[针对第三方身份提供商创建角色 \(联合身份验证 \)](#)的说明进行操作。

- IAM 用户：

- 创建您的用户可以担任的角色。按照《IAM 用户指南》中[为 IAM 用户创建角色](#)的说明进行操作。
- (不推荐使用) 将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中[向用户添加权限 \(控制台 \)](#)中的说明进行操作。

步骤 3：在 S3 桶上设置权限

默认情况下，所有 S3 桶和对象都是私有的。仅资源所有者 (创建了桶的 Amazon Web Services 账户) 能够访问桶及其包含的任何对象。不过，资源所有者可以选择通过编写访问策略来向其他资源和用户授予访问权限。

当您设置策略时，建议包含一个随机生成的字符串作为存储桶的前缀，这样只会将目标日志流导出到该存储桶。

Important

为了使导出到 S3 桶更加安全，我们现在要求您指定允许将日志数据导出到 S3 桶的源账户列表。

在以下示例中，`aws:SourceAccount` 密钥中账户 ID 的列表会是用户可以从其中将日志数据导出到 S3 存储桶的账户。`aws:SourceArn` 密钥是正在进行的操作的资源。如本示例所示，您可以将其限制为特定的日志组，也可以使用通配符。

建议您还包括创建 S3 桶的账户的账户 ID，以允许在同一账户内导出。

设置 Amazon S3 存储桶上的权限

1. 在 Amazon S3 控制台中，选择您在第 1 步中创建的存储桶。
2. 选择 Permissions (权限)、Bucket policy (存储桶策略)。
3. 在 Bucket Policy Editor (桶策略编辑器) 中，添加以下策略。将 `my-exported-logs` 更改为 S3 存储桶的名称。请务必指定正确的区域端点，例如 Principal (委托人) 的 `us-west-1`。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "s3:GetBucketAcl",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": [
            "123456789012",
            "111122223333"
          ]
        }
      },
      "ArnLike": {
        "aws:SourceArn": [
```

```

        "arn:aws:logs:us-east-1:123456789012:log-group:*",
        "arn:aws:logs:us-east-1:111122223333:log-group:*"
    ]
}
},
{
    "Action": "s3:PutObject",
    "Effect": "Allow",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
    "Condition": {
        "StringEquals": {
            "s3:x-amz-acl": "bucket-owner-full-control",
            "aws:SourceAccount": [
                "123456789012",
                "111122223333"
            ]
        }
    },
    "ArnLike": {
        "aws:SourceArn": [
            "arn:aws:logs:us-east-1:123456789012:log-group:*",
            "arn:aws:logs:us-east-1:111122223333:log-group:*"
        ]
    }
},
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/role_name"
    },
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition": {
        "StringEquals": {
            "s3:x-amz-acl": "bucket-owner-full-control"
        }
    }
}
]
}

```

4. 选择 **Save**，将您刚添加的策略设置为存储桶上的访问策略。此策略可让 CloudWatch Logs 将日志数据导出到您的 S3 桶中。存储桶所有者对所有导出的对象拥有完全权限。

 Warning

如果现有存储桶已附加一个或多个策略，请将 CloudWatch Logs 访问的语句添加到这些策略。我们建议您评估生成的权限集，确保它们适合访问存储桶的用户。

(可选) 步骤 4：导出到使用 SSE-KMS 加密的桶

仅当您要导出到使用服务器端通过 Amazon KMS keys 加密的 S3 桶时，才需要执行此步骤。这种加密称为 SSE-KMS。

导出到使用 SSE-KMS 加密的桶

1. 从 <https://console.aws.amazon.com/kms> 打开 Amazon KMS 控制台。
2. 要更改 Amazon Web Services 区域，请使用页面右上角的区域选择器。
3. 在左侧导航栏中，选择 **Customer managed keys (客户托管密钥)**。

选择 **Create Key (创建密钥)**。
4. 对于 **Key type (密钥类型)**，选择 **Symmetric (对称)**。
5. 对于 **Key usage (密钥用法)**，选择 **Encrypt and decrypt (加密和解密)**，然后选择 **Next (下一步)**。
6. 在 **Add labels (添加标签)** 下，输入密钥的别名，并添加描述或标签 (可选)。然后选择下一步。
7. 在 **Key administrators (密钥管理员)** 下，选择可以管理此密钥的人员，然后选择 **Next (下一步)**。
8. 在 **Define key usage permissions (定义密钥使用权限)** 下，不进行任何更改并选择 **Next (下一步)**。
9. 检查设置，然后选择 **Finish (完成)**。
10. 返回 **Customer managed keys (客户托管密钥)** 页面，选择您刚刚创建的密钥的名称。
11. 选择 **Key policy (密钥策略)** 选项卡，然后选择 **Switch to policy view (切换到策略视图)**。
12. 在 **Key policy (密钥策略)** 部分，选择 **Edit (编辑)**。
13. 将以下语句添加到密钥策略语句列表。执行此操作时，将 *Region* 替换为日志的区域，将 *account-ARN* 替换为拥有 KMS 密钥的账户的 ARN。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow CWL Service Principal usage",
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "account-ARN"
      },
      "Action": [
        "kms:GetKeyPolicy*",
        "kms:PutKeyPolicy*",
        "kms:DescribeKey*",
        "kms:CreateAlias*",
        "kms:ScheduleKeyDeletion*",
        "kms:Decrypt"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Enable IAM Role Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/role_name"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ]
    }
  ]
}
```

```
    ],  
    "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-id"  
  }  
]  
}
```

14. 选择保存更改。
15. 通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
16. 找到您在 [步骤 1：创建 S3 存储桶](#) 中创建的桶并选择桶名称。
17. 选择属性选项卡。然后，在 Default Encryption (默认加密) 下，选择 Edit (编辑)。
18. 在 Server-side Encryption (服务器端加密) 下，选择 Enable (启用)。
19. 在 Encryption type (加密类型) 下，请选择 Amazon Key Management Service key (SSE-KMS) (Amazon Key Management Service 密钥(SSE-KMS))。
20. 选择 Choose from your Amazon KMS keys (从密钥中选择)，然后找到您创建的密钥。
21. 对于 Bucket Key (桶密钥)，选择 Enable (启用)。
22. 选择保存更改。

步骤 5：创建导出任务

在本步骤中，您可创建导出任务以便从日志组中导出日志。

使用 CloudWatch 控制台将数据导出到 Amazon S3

1. 如 [步骤 2：设置访问权限](#) 中所述，使用足够的权限登录。
2. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
3. 在导航窗格中，选择 Log groups (日志组)。
4. 在 Log Groups (日志组) 屏幕上，选择日志组的名称。
5. 选择 Actions (操作)、Export data to Amazon S3 (将数据导出到 Amazon S3)。
6. 在 Export data to Amazon S3 (将数据导出到 Amazon S3) 屏幕的 Define data export (定义数据导出) 下，使用 From (从) 和 To (至) 设置要导出的数据的时间范围。
7. 如果日志组有多个日志流，您可以提供一个日志流前缀，将日志组数据限定为特定流。选择 Advanced (高级)，然后为 Stream prefix (流前缀) 输入日志流前缀。
8. 在 Choose S3 bucket (选择 S3 桶) 下，选择与 S3 桶关联的账户。
9. 对于 S3 bucket name (S3 桶名称)，选择 S3 桶。

10. 对于 S3 Bucket prefix (S3 存储桶前缀) , 输入在存储桶策略中指定的随机生成的字符串。
11. 选择 Export (导出) , 将日志数据导出到 Amazon S3。
12. 要查看您导出到 Amazon S3 的日志数据的状态, 请依次选择 Actions (操作) 、 View all exports to Amazon S3 (查看导出到 Amazon S3 的所有内容) 。

使用 Amazon CLI 将日志数据导出到 Amazon S3

在以下示例中, 您将使用导出任务将所有数据从名为 my-log-group 的 CloudWatch Logs 日志组导出到名为 my-exported-logs 的 Amazon S3 存储桶中。此示例假定您已创建了一个名为 my-log-group 的日志组。

支持将日志数据导出到由 Amazon KMS 加密的 S3 桶。不支持导出到由 DSSE-KMS 加密的存储桶。

有关如何设置导出的详细信息取决于您要导出到的 Amazon S3 存储桶与要导出的日志是位于同一个账户中, 还是不同的账户中。

主题

- [同账号导出](#)
- [跨账户导出](#)

同账号导出

如果 Amazon S3 存储桶与要导出的日志位于同一个账户中, 请参阅本节中的说明。

主题

- [步骤 1 : 创建 S3 存储桶](#)
- [步骤 2 : 设置访问权限](#)
- [步骤 3 : 在 S3 桶上设置权限](#)
- [\(可选 \) 步骤 4 : 导出到使用 SSE-KMS 加密的桶](#)
- [步骤 5 : 创建导出任务](#)

步骤 1 : 创建 S3 存储桶

我们建议您使用专为 CloudWatch Logs 创建的存储桶。但是, 如果要使用现有存储桶, 请跳至第 2 步。

Note

S3 桶必须与要导出的日志数据位于同一区域。CloudWatch Logs 不支持将数据导出到其他区域中的 S3 桶。

使用 Amazon CLI 创建 S3 桶

在命令提示符处，运行以下 [create-bucket](#) 命令，其中 LocationConstraint 是您要将日志数据导出到的区域。

```
aws s3api create-bucket --bucket my-exported-logs --create-bucket-configuration  
LocationConstraint=us-east-2
```

下面是示例输出。

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

步骤 2：设置访问权限

要在步骤 5 中创建导出任务，您需要分配 AmazonS3ReadOnlyAccess IAM 角色和以下权限：

- logs:CreateExportTask
- logs:CancelExportTask
- logs:DescribeExportTasks
- logs:DescribeLogStreams
- logs:DescribeLogGroups

要提供访问权限，请为您的用户、组或角色添加权限：

- 通过身份提供商在 IAM 中托管的用户：

创建适用于身份联合验证的角色。按照《IAM 用户指南》中[针对第三方身份提供商创建角色 \(联合身份验证\)](#)的说明进行操作。

- IAM 用户：
 - 创建您的用户可以担任的角色。按照《IAM 用户指南》中[为 IAM 用户创建角色](#)的说明进行操作。

- (不推荐使用) 将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中[向用户添加权限 \(控制台\)](#)中的说明进行操作。

步骤 3：在 S3 桶上设置权限

默认情况下，所有 S3 桶和对象都是私有的。仅资源所有者（创建了存储桶的账户）能够访问存储桶及其包含的任何对象。不过，资源所有者可以选择通过编写访问策略来向其他资源和用户授予访问权限。

Important

为了使导出到 S3 桶更加安全，我们现在要求您指定允许将日志数据导出到 S3 桶的源账户列表。

在以下示例中，aws:SourceAccount 密钥中账户 ID 的列表会是用户可以从其中将日志数据导出到 S3 存储桶的账户。aws:SourceArn 密钥是正在进行的操作的资源。如本示例所示，您可以将其限制为特定的日志组，也可以使用通配符。

建议您还包括创建 S3 桶的账户的账户 ID，以允许在同一账户内导出。

在 S3 桶上设置权限

1. 创建一个名为 policy.json 的文件并添加以下访问策略，将 my-exported-logs 更改为您的 S3 存储桶的名称，并将 Principal 更改为您要将日志数据导出到的区域的端点，例如 us-west-1。使用文本编辑器以创建此策略文件。请勿使用 IAM 控制台。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGetBucketAcl",
      "Action": "s3:GetBucketAcl",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": [
```

```

        "123456789012",
        "111122223333"
    ]
},
"ArnLike": {
    "aws:SourceArn": [
        "arn:aws:logs:us-east-1:123456789012:log-group:*",
        "arn:aws:logs:us-east-1:111122223333:log-group:*"
    ]
}
},
{
    "Sid": "AllowPutObject",
    "Action": "s3:PutObject",
    "Effect": "Allow",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
    "Condition": {
        "StringEquals": {
            "s3:x-amz-acl": "bucket-owner-full-control",
            "aws:SourceAccount": [
                "123456789012",
                "111122223333"
            ]
        }
    },
    "ArnLike": {
        "aws:SourceArn": [
            "arn:aws:logs:us-east-1:123456789012:log-group:*",
            "arn:aws:logs:us-east-1:111122223333:log-group:*"
        ]
    }
}
}
]
}

```

2. 使用 [put-bucket-policy](#) 命令将您刚添加的策略设置为存储桶上的访问策略。此策略可让 CloudWatch Logs 将日志数据导出到您的 S3 桶中。存储桶所有者将对所有导出的对象拥有完全权限。

```
aws s3api put-bucket-policy --bucket my-exported-logs --policy file://policy.json
```

⚠ Warning

如果现有存储桶已附加一个或多个策略，请将 CloudWatch Logs 访问的语句添加到这些策略。我们建议您评估生成的权限集，确保它们适合访问存储桶的用户。

(可选) 步骤 4：导出到使用 SSE-KMS 加密的桶

仅当您要导出到使用服务器端通过 Amazon KMS keys 加密的 S3 桶时，才需要执行此步骤。这种加密称为 SSE-KMS。

导出到使用 SSE-KMS 加密的桶

1. 使用文本编辑器创建名为 `key_policy.json` 的文件，并添加以下访问策略。添加策略时，进行以下更改：
 - 将 *Region* 替换为日志的区域。
 - 将 *account-ARN* 替换为拥有 KMS 密钥的账户的 ARN。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow CWL Service Principal usage",
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.Region.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
```

```
    "Principal": {
      "AWS": "account-ARN"
    },
    "Action": [
      "kms:GetKeyPolicy*",
      "kms:PutKeyPolicy*",
      "kms:DescribeKey*",
      "kms:CreateAlias*",
      "kms:ScheduleKeyDeletion*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }
}
```

2. 输入以下命令：

```
aws kms create-key --policy file://key_policy.json
```

下面是此命令的示例输出：

```
{
  "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. 使用文本编辑器创建名为 bucketencryption.json 的文件，其中包含以下内容。

```
{
  "Rules": [
    {
      "ApplyServerSideEncryptionByDefault": {
        "SSEAlgorithm": "aws:kms",
        "KMSEMasterKeyID": "{KMS Key ARN}"
      },
      "BucketKeyEnabled": true
    }
  ]
}
```

4. 输入以下命令，将 *bucket-name* 替换为您要将日志导出到的桶的名称。

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

如果该命令没有返回错误，则该过程成功。

步骤 5：创建导出任务

使用以下命令创建导出任务。创建后，导出任务可能需要几秒到几小时的时间才能完成，具体取决于要导出的数据大小。

使用 Amazon CLI 将数据导出到 Amazon S3。

1. 如 [步骤 2：设置访问权限](#) 中所述，使用足够的权限登录。
2. 在命令提示符处，使用以下 [create-export-task](#) 命令创建导出任务。

```
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"
```

下面是示例输出。

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

跨账户导出

如果 Amazon S3 存储桶与要导出的日志位于不同账户中，请参阅本节中的说明。

主题

- [步骤 1：创建 S3 存储桶](#)
- [步骤 2：设置访问权限](#)
- [步骤 3：在 S3 桶上设置权限](#)
- [\(可选 \) 步骤 4：导出到使用 SSE-KMS 加密的桶](#)
- [步骤 5：创建导出任务](#)

步骤 1：创建 S3 存储桶

我们建议您使用专为 CloudWatch Logs 创建的存储桶。但是，如果要使用现有存储桶，请跳至第 2 步。

Note

S3 桶必须与要导出的日志数据位于同一区域。CloudWatch Logs 不支持将数据导出到其他区域中的 S3 桶。

使用 Amazon CLI 创建 S3 桶

在命令提示符处，运行以下 [create-bucket](#) 命令，其中 LocationConstraint 是您要将日志数据导出到的区域。

```
aws s3api create-bucket --bucket my-exported-logs --create-bucket-configuration
LocationConstraint=us-east-2
```

下面是示例输出。

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

步骤 2：设置访问权限

首先，您必须创建新的 IAM policy，以使 CloudWatch Logs 拥有目标 Amazon S3 存储桶的 `s3:PutObject` 权限。

要在步骤 5 中创建导出任务，您需要使用 `AmazonS3ReadOnlyAccess` IAM 角色登录并且具有其他某些权限。您可以创建包含其他必要权限的策略。

您创建的策略取决于目标存储桶是否使用 Amazon KMS 加密。如果不使用 Amazon KMS 加密，请创建包含以下内容的策略。

JSON

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

如果目标存储桶使用 Amazon KMS 加密，请创建包含以下内容的策略。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::>amzn-s3-demo-bucket/*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "kms:GenerateDataKey",
      "kms:Decrypt"
    ]
  }
]
```



```
    ],
    "Resource": "arn:aws:kms:us-  
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"  
  }  
]  
}
```

要在步骤 5 中创建导出任务，您必须使用 AmazonS3ReadOnlyAccess IAM 角色登录，并且使用刚刚创建的 IAM policy 且具有以下权限：

- logs:CreateExportTask
- logs:CancelExportTask
- logs:DescribeExportTasks
- logs:DescribeLogStreams
- logs:DescribeLogGroups

要提供访问权限，请为您的用户、组或角色添加权限：

- 通过身份提供商在 IAM 中托管的用户：

创建适用于身份联合验证的角色。按照《IAM 用户指南》中[针对第三方身份提供商创建角色 \(联合身份验证 \)](#)的说明进行操作。

- IAM 用户：

- 创建您的用户可以担任的角色。按照《IAM 用户指南》中[为 IAM 用户创建角色](#)的说明进行操作。
- (不推荐使用) 将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中[向用户添加权限 \(控制台 \)](#)中的说明进行操作。

步骤 3：在 S3 桶上设置权限

默认情况下，所有 S3 桶和对象都是私有的。仅资源所有者 (创建了存储桶的账户) 能够访问存储桶及其包含的任何对象。不过，资源所有者可以选择通过编写访问策略来向其他资源和用户授予访问权限。

Important

为了使导出到 S3 桶更加安全，我们现在要求您指定允许将日志数据导出到 S3 桶的源账户列表。

在以下示例中，aws:SourceAccount 密钥中账户 ID 的列表会是用户可以从其中将日志数据导出到 S3 存储桶的账户。aws:SourceArn 密钥是正在进行的操作的资源。如本示例所示，您可以将其限制为特定的日志组，也可以使用通配符。建议您还包括创建 S3 桶的账户的账户 ID，以允许在同一账户内导出。

在 S3 桶上设置权限

1. 创建一个名为 policy.json 的文件并添加以下访问策略，将 my-exported-logs 更改为您的 S3 存储桶的名称，并将 Principal 更改为您要将日志数据导出到的区域的端点，例如 us-west-1。使用文本编辑器以创建此策略文件。请勿使用 IAM 控制台。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "s3:GetBucketAcl",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": [
            "123456789012",
            "111122223333"
          ]
        }
      },
      "ArnLike": {
        "aws:SourceArn": [
          "arn:aws:logs:us-east-1:123456789012:log-group:*",
          "arn:aws:logs:us-east-1:111122223333:log-group:*"
        ]
      }
    },
    {
      "Action": "s3:PutObject",
      "Effect": "Allow",
```

```

    "Resource": "arn:aws:s3:::my-exported-logs/*",
    "Principal": { "Service": "logs.Region.amazonaws.com" },
    "Condition": {
      "StringEquals": {
        "s3:x-amz-acl": "bucket-owner-full-control",
        "aws:SourceAccount": [
          "123456789012",
          "111122223333"
        ]
      },
      "ArnLike": {
        "aws:SourceArn": [
          "arn:aws:logs:us-east-1:123456789012:log-group:*",
          "arn:aws:logs:us-east-1:111122223333:log-group:*"
        ]
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/role_name"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::>amzn-s3-demo-bucket/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    }
  ]
}

```

2. 使用 [put-bucket-policy](#) 命令将您刚添加的策略设置为存储桶上的访问策略。此策略可让 CloudWatch Logs 将日志数据导出到您的 S3 桶中。存储桶所有者将对所有导出的对象拥有完全权限。

```
aws s3api put-bucket-policy --bucket my-exported-logs --policy file://policy.json
```

⚠ Warning

如果现有存储桶已附加一个或多个策略，请将 CloudWatch Logs 访问的语句添加到这些策略。我们建议您评估生成的权限集，确保它们适合访问存储桶的用户。

(可选) 步骤 4：导出到使用 SSE-KMS 加密的桶

仅当您要导出到使用服务器端通过 Amazon KMS keys 加密的 S3 桶时，才需要执行此步骤。这种加密称为 SSE-KMS。

导出到使用 SSE-KMS 加密的桶

1. 使用文本编辑器创建名为 `key_policy.json` 的文件，并添加以下访问策略。添加策略时，进行以下更改：
 - 将 *Region* 替换为日志的区域。
 - 将 *account-ARN* 替换为拥有 KMS 密钥的账户的 ARN。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCWLSERVICEPrincipalUsage",
      "Effect": "Allow",
      "Principal": {
        "Service": "logs.us-east-1.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*"
    },
    {
      "Sid": "EnableIAMUserPermissions",
```

```

        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:root"
        },
        "Action": [
            "kms:GetKeyPolicy*",
            "kms:PutKeyPolicy*",
            "kms:DescribeKey*",
            "kms:CreateAlias*",
            "kms:ScheduleKeyDeletion*",
            "kms:Decrypt"
        ],
        "Resource": "*"
    },
    {
        "Sid": "EnableIAMRolePermissions",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/role_name"
        },
        "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
        ],
        "Resource": "arn:aws:kms:us-east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
]
}

```

2. 输入以下命令：

```
aws kms create-key --policy file://key_policy.json
```

下面是此命令的示例输出：

```

{
  "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. 使用文本编辑器创建名为 `bucketencryption.json` 的文件，其中包含以下内容。

```
{
  "Rules": [
    {
      "ApplyServerSideEncryptionByDefault": {
        "SSEAlgorithm": "aws:kms",
        "KMSMasterKeyID": "{KMS Key ARN}"
      },
      "BucketKeyEnabled": true
    }
  ]
}
```

4. 输入以下命令，将 *bucket-name* 替换为您要将日志导出到的桶的名称。

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

如果该命令没有返回错误，则该过程成功。

步骤 5：创建导出任务

使用以下命令创建导出任务。创建后，导出任务可能需要几秒到几小时的时间才能完成，具体取决于要导出的数据大小。

使用 Amazon CLI 将数据导出到 Amazon S3。

1. 如 [步骤 2：设置访问权限](#) 中所述，使用足够的权限登录。

2. 在命令提示符处，使用以下 [create-export-task](#) 命令创建导出任务。

```
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"
```

下面是示例输出。

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

描述导出任务

创建导出任务后，您可以获取任务的当前状态。

使用 Amazon CLI 描述导出任务

在命令提示符处，使用以下 [describe-export-tasks](#) 命令。

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

下面是示例输出。

```
{
  "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
  }]
}

```

您可以三种不同的方式使用 `describe-export-tasks` 命令：

- 无任何筛选器 - 按照与创建时间相反的顺序列出所有导出任务。
- 筛选任务 ID - 仅列出具有指定 ID 的导出任务（如果有）。
- 筛选任务状态 - 列出具有指定状态的导出任务。

例如，使用以下命令筛选 FAILED 状态。

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

下面是示例输出。

```

{
  "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
    }
  ]
}

```


取消导出任务

您可以取消处于 PENDING 或 RUNNING 状态的导出任务。

使用 Amazon CLI 取消导出任务

在命令提示符处，使用以下 [cancel-export-task](#) 命令：

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

您可以使用 [describe-export-tasks](#) 命令验证任务是否已成功取消。

将 CloudWatch Logs 数据流式传输到 Amazon OpenSearch Service

您可以在 Amazon CloudWatch Logs 中配置日志组，以便可以近乎实时地将数据流式传输到 Amazon OpenSearch Service 集群。有关更多信息，请参阅 [使用订阅实时处理日志数据](#)。

Note

仅标准日志类中的日志组支持流式传输到 OpenSearch 服务。有关日志类的更多信息，请参阅 [日志类](#)。

根据流式传输的日志数据量，请考虑设置函数级并发限制。有关更多信息，请参阅 [Lambda 函数扩展](#)。

Note

由于将大量 CloudWatch Logs 数据流式传输到 OpenSearch Service 可能会导致高额使用费，因此我们建议您在 Amazon 账单与成本管理 控制台中创建预算。有关更多信息，请参阅[通过 Amazon Budgets 管理成本](#)。

本节介绍在将日志组订阅到 OpenSearch Service 之前必须完成的先决条件。它还介绍了如何将日志组订阅到 OpenSearch Service。

先决条件

您需要首先创建一个 OpenSearch Service 域，然后才能开始此操作。该域可以具有公有访问权限或 VPC 访问权限，但您无法在创建该域后修改访问权限的类型。之后您可能需要检查 OpenSearch Service 域设置，并根据集群将会处理的数据量修改集群配置。有关创建域的说明，请参阅[创建 OpenSearch Service 域](#)。

有关 OpenSearch Service 的更多信息，请参阅《Amazon OpenSearch Service 开发人员指南》<https://docs.amazonaws.cn/opensearch-service/latest/developerguide/>。

将日志组订阅到 OpenSearch Service

您可以通过 CloudWatch 控制台将日志组订阅到 OpenSearch Service。

将日志组订阅到 OpenSearch Service

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Log groups (日志组)。
3. 选择日志组的名称。
4. 选择 Actions (操作)、Subscription filters (订阅筛选器)、Create Amazon OpenSearch Service subscription filter (创建 Amazon OpenSearch Service 订阅筛选器)。
5. 选择是要流式传输到此账户还是其他账户中的集群。
 - 如果您选择了此账户，请选择上一步中已创建的域。
 - 如果您选择了其他账户，请提供域 ARN 和端点。
6. 对于 Lambda IAM Execution Role (Lambda IAM 执行角色)，请选择 Lambda 在执行对 OpenSearch 的调用时应使用的 IAM 角色。

您选择的 IAM 角色必须满足以下要求：

- 它在信任关系中必须具有 `lambda.amazonaws.com`。
- 它必须包含以下策略：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowOpenSearchStreamingAccess",
      "Action": [
        "es:*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:es:us-east-1:123456789012:domain/cloudwatch-logs/*"
    }
  ]
}
```

- 如果目标 OpenSearch Service 域使用 VPC 访问权限，则该角色必须已附加了 AWSLambdaVPCAccessExecutionRole 策略。此 Amazon 托管式策略将向 Lambda 授予访问客户 VPC 的权限，从而确保 Lambda 能够写入 VPC 中的 OpenSearch 终端节点。
7. 对于 Log format (日志格式)，请选择日志格式。
 8. 对于 Subscription filter pattern (订阅筛选条件模式)，键入要在您的日志事件中查找的字词或模式。这可确保只将您需要的数据发送到 OpenSearch 集群。有关更多信息，请参阅 [使用筛选条件从日志事件创建指标](#)。
 9. (可选) 对于 Select log data to test (选择要测试的日志数据)，请选择一个日志流，然后选择 Test pattern (测试模式)，以确认搜索筛选器是否会返回您期望的结果。
 10. 选择 Start streaming (开始流式传输)。

使用 Amazon SDK 的 CloudWatch Logs 代码示例

以下代码示例展示如何将 CloudWatch Logs 与 Amazon 软件开发工具包 (SDK) 结合使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 Amazon Web Services 服务 服务结合来完成特定任务的代码示例。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

- [使用 Amazon SDK 的 CloudWatch Logs 基础知识示例](#)
 - [使用 Amazon SDK 的 CloudWatch Logs 操作](#)
 - [将 AssociateKmsKey 和 Amazon SDK 搭配使用](#)
 - [将 CancelExportTask 和 Amazon SDK 搭配使用](#)
 - [将 CreateExportTask 和 Amazon SDK 搭配使用](#)
 - [将 CreateLogGroup 与 Amazon SDK 或 CLI 配合使用](#)
 - [将 CreateLogStream 与 Amazon SDK 或 CLI 配合使用](#)
 - [将 DeleteLogGroup 与 Amazon SDK 或 CLI 配合使用](#)
 - [将 DeleteSubscriptionFilter 和 Amazon SDK 搭配使用](#)
 - [将 DescribeExportTasks 和 Amazon SDK 搭配使用](#)
 - [将 DescribeLogGroups 与 Amazon SDK 或 CLI 配合使用](#)
 - [将 DescribeLogStreams 与 Amazon SDK 或 CLI 配合使用](#)
 - [将 DescribeSubscriptionFilters 和 Amazon SDK 搭配使用](#)
 - [将 GetLogEvents 与 Amazon SDK 或 CLI 配合使用](#)
 - [将 GetQueryResults 和 Amazon SDK 搭配使用](#)
 - [将 PutSubscriptionFilter 和 Amazon SDK 搭配使用](#)
 - [将 StartLiveTail 和 Amazon SDK 搭配使用](#)
 - [将 StartQuery 和 Amazon SDK 搭配使用](#)
- [使用 Amazon SDK 的 CloudWatch Logs 场景](#)
 - [使用 CloudWatch Logs 运行大型查询](#)

- [使用计划的事件调用 Lambda 函数](#)

使用 Amazon SDK 的 CloudWatch Logs 基础知识示例

以下代码示例展示了如何将 Amazon CloudWatch Logs 的基本功能与 Amazon SDK 结合使用。

示例

- [使用 Amazon SDK 的 CloudWatch Logs 操作](#)
 - [将 AssociateKmsKey 和 Amazon SDK 搭配使用](#)
 - [将 CancelExportTask 和 Amazon SDK 搭配使用](#)
 - [将 CreateExportTask 和 Amazon SDK 搭配使用](#)
 - [将 CreateLogGroup 与 Amazon SDK 或 CLI 配合使用](#)
 - [将 CreateLogStream 与 Amazon SDK 或 CLI 配合使用](#)
 - [将 DeleteLogGroup 与 Amazon SDK 或 CLI 配合使用](#)
 - [将 DeleteSubscriptionFilter 和 Amazon SDK 搭配使用](#)
 - [将 DescribeExportTasks 和 Amazon SDK 搭配使用](#)
 - [将 DescribeLogGroups 与 Amazon SDK 或 CLI 配合使用](#)
 - [将 DescribeLogStreams 与 Amazon SDK 或 CLI 配合使用](#)
 - [将 DescribeSubscriptionFilters 和 Amazon SDK 搭配使用](#)
 - [将 GetLogEvents 与 Amazon SDK 或 CLI 配合使用](#)
 - [将 GetQueryResults 和 Amazon SDK 搭配使用](#)
 - [将 PutSubscriptionFilter 和 Amazon SDK 搭配使用](#)
 - [将 StartLiveTail 和 Amazon SDK 搭配使用](#)
 - [将 StartQuery 和 Amazon SDK 搭配使用](#)

使用 Amazon SDK 的 CloudWatch Logs 操作

以下代码示例演示了如何使用 Amazon SDK 来执行单个 CloudWatch Logs 操作。每个示例都包含一个指向 GitHub 的链接，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 CloudWatch Logs API，是必须在上下文中运行的大型程序的代码节选。您可以在[使用 Amazon SDK 的 CloudWatch Logs 场景](#)中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅[Amazon CloudWatch Logs API 参考](#)。

示例

- [将 AssociateKmsKey 和 Amazon SDK 搭配使用](#)
- [将 CancelExportTask 和 Amazon SDK 搭配使用](#)
- [将 CreateExportTask 和 Amazon SDK 搭配使用](#)
- [将 CreateLogGroup 与 Amazon SDK 或 CLI 配合使用](#)
- [将 CreateLogStream 与 Amazon SDK 或 CLI 配合使用](#)
- [将 DeleteLogGroup 与 Amazon SDK 或 CLI 配合使用](#)
- [将 DeleteSubscriptionFilter 和 Amazon SDK 搭配使用](#)
- [将 DescribeExportTasks 和 Amazon SDK 搭配使用](#)
- [将 DescribeLogGroups 与 Amazon SDK 或 CLI 配合使用](#)
- [将 DescribeLogStreams 与 Amazon SDK 或 CLI 配合使用](#)
- [将 DescribeSubscriptionFilters 和 Amazon SDK 搭配使用](#)
- [将 GetLogEvents 与 Amazon SDK 或 CLI 配合使用](#)
- [将 GetQueryResults 和 Amazon SDK 搭配使用](#)
- [将 PutSubscriptionFilter 和 Amazon SDK 搭配使用](#)
- [将 StartLiveTail 和 Amazon SDK 搭配使用](#)
- [将 StartQuery 和 Amazon SDK 搭配使用](#)

将 **AssociateKmsKey** 和 Amazon SDK 搭配使用

以下代码示例演示了如何使用 AssociateKmsKey。

.NET

适用于 .NET 的 Amazon SDK

Note

查看 [GitHub](#)，了解更多信息。查找完整示例，了解如何在 [Amazon 代码示例存储库](#) 中进行设置和运行。

```
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}.");
        }
    }
}
```


- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API 参考》中的 [AssociateKmsKey](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CancelExportTask** 和 Amazon SDK 搭配使用

以下代码示例演示了如何使用 `CancelExportTask`。

.NET

适用于 .NET 的 Amazon SDK

Note

查看 [GitHub](#)，了解更多信息。查找完整示例，了解如何在 [Amazon 代码示例存储库](#) 中进行设置和运行。

```
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.");
    }
}
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API 参考》中的 [CancelExportTask](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateExportTask** 和 Amazon SDK 搭配使用

以下代码示例演示了如何使用 `CreateExportTask`。

.NET

适用于 .NET 的 Amazon SDK

Note

查看 GitHub，了解更多信息。查找完整示例，了解如何在 [Amazon 代码示例存储库](#) 中进行设置和运行。

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
```

```

using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Export Task to export the contents of the Amazon
/// CloudWatch Logs to the specified Amazon Simple Storage Service (Amazon
S3)
/// bucket.
/// </summary>
public class CreateExportTask
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string taskName = "export-task-example";
        string logGroupName = "cloudwatchlogs-example-loggroup";
        string destination = "amzn-s3-demo-bucket";
        var fromTime = 1437584472382;
        var toTime = 1437584472833;

        var request = new CreateExportTaskRequest
        {
            From = fromTime,
            To = toTime,
            TaskName = taskName,
            LogGroupName = logGroupName,
            Destination = destination,
        };

        var response = await client.CreateExportTaskAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"The task, {taskName} with ID: " +
                              $"{response.TaskId} has been created
successfully.");
        }
    }
}

```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API 参考》中的 [CreateExportTask](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateLogGroup** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateLogGroup。

.NET

适用于 .NET 的 Amazon SDK

Note

查看 GitHub，了解更多信息。查找完整示例，了解如何在 [Amazon 代码示例存储库](#) 中进行设置和运行。

```
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.");
}
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API 参考》中的 [CreateLogGroup](#)。

CLI

Amazon CLI

以下命令将创建名为 my-logs 的日志组：

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

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 [CreateLogGroup](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 [CreateLogGroup](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateLogStream** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateLogStream`。

.NET

适用于 .NET 的 Amazon SDK

Note

查看 GitHub，了解更多信息。查找完整示例，了解如何在 [Amazon 代码示例存储库](#) 中进行设置和运行。

```
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.");  
    }  
}  
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API 参考》中的 [CreateLogStream](#)。

CLI

Amazon CLI

以下命令将在日志组 my-logs 中创建一个名为 20150601 的日志流：

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

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 [CreateLogStream](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 DeleteLogGroup 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteLogGroup。

.NET

适用于 .NET 的 Amazon SDK

Note

查看 GitHub，了解更多信息。查找完整示例，了解如何在 [Amazon 代码示例存储库](#) 中进行设置和运行。


```
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}.");
        }
    }
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API 参考》中的 [DeleteLogGroup](#)。

CLI

Amazon CLI

以下命令将删除名为 my-logs 的日志组：

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

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 [DeleteLogGroup](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 [DeleteLogGroup](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 `DeleteSubscriptionFilter` 和 Amazon SDK 搭配使用

以下代码示例演示如何使用 `DeleteSubscriptionFilter`。

C++

SDK for C++

Note

查看 [GitHub](#)，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

包含所需的文件。

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

删除订阅筛选条件。

```
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;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API 参考》中的 [DeleteSubscriptionFilter](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [DeleteSubscriptionFilter](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 [DeleteSubscriptionFilter](#)。

SDK for JavaScript (v2)

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
// 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);
    }
});
```

- 有关更多信息，请参阅 [《适用于 JavaScript 的 Amazon SDK 开发人员指南》](#)。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 [DeleteSubscriptionFilter](#)。

Kotlin

SDK for Kotlin

Note

查看 GitHub，了解更多信息。查找完整示例，了解如何在 [Amazon 代码示例存储库](#) 中进行设置和运行。

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

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

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

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [DeleteSubscriptionFilter](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeExportTasks** 和 Amazon SDK 搭配使用

以下代码示例演示了如何使用 DescribeExportTasks。

.NET

适用于 .NET 的 Amazon SDK

Note

查看 GitHub，了解更多信息。查找完整示例，了解如何在 [Amazon 代码示例存储库](#) 中进行设置和运行。

```
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);
        }
    }
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API 参考》中的 [DescribeExportTasks](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeLogGroups** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeLogGroups。

.NET

适用于 .NET 的 Amazon SDK

Note

查看 [GitHub](#)，了解更多信息。查找完整示例，了解如何在 [Amazon 代码示例存储库](#) 中进行设置和运行。

```
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);
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API 参考》中的 [DescribeLogGroups](#)。

CLI

Amazon CLI

以下命令将描述名为 my-logs 的日志组：

```
aws logs describe-log-groups --log-group-name-prefix my-logs
```

输出：

```
{
  "logGroups": [
    {
      "storedBytes": 0,
      "metricFilterCount": 0,
      "creationTime": 1433189500783,
      "logGroupName": "my-logs",
      "retentionInDays": 5,
      "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:*"
    }
  ]
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 [DescribeLogGroups](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
import {
  paginateDescribeLogGroups,
  CloudWatchLogsClient,
} from "@aws-sdk/client-cloudwatch-logs";

const client = new CloudWatchLogsClient({});

export const main = async () => {
  const paginatedLogGroups = paginateDescribeLogGroups({ client }, {});
  const logGroups = [];

  for await (const page of paginatedLogGroups) {
```

```
    if (page.logGroups?.every((lg) => !!lg)) {
      logGroups.push(...page.logGroups);
    }
  }

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

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 [DescribeLogGroups](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeLogStreams** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeLogStreams。

CLI

Amazon CLI

以下命令显示日志组 my-logs 中以前缀 2015 开头的所有日志流：

```
aws logs describe-log-streams --log-group-name my-logs --log-stream-name-  
prefix 2015
```

输出：

```
{
  "logStreams": [
    {
      "creationTime": 1433189871774,
      "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-  
stream:20150531",
      "logStreamName": "20150531",
      "storedBytes": 0
    },
    {
      "creationTime": 1433189873898,
```

```
        "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-
stream:20150601",
        "logStreamName": "20150601",
        "storedBytes": 0
    }
]
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 [DescribeLogStreams](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

在指定日志组中搜索与给定前缀匹配的日志流。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class CloudWatchLogsSearch {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <logGroupName> <logStreamName>

            Where:
```

```

        logGroupName - The name of the log group (for example,
        WeathertopJavaContainerLogs).
        logStreamName - The name of the log stream (for example,
        weathertop-java-stream).
        pattern - the pattern to use (for example, INFO)

        """;

    if (args.length != 3) {
        System.out.print(usage);
        System.exit(1);
    }

    String logGroupName = args[0] ;
    String logStreamName = args[1] ;
    String pattern = args[2] ;

    CloudWatchLogsClient cwlClient = CloudWatchLogsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    searchLogStreamsAndFilterEvents(cwlClient, logGroupName, logStreamName,
    pattern);
}

/**
 * Searches for log streams with a specific prefix within a log group and
 * filters log events based on a specified pattern.
 *
 * @param cwlClient      the CloudWatchLogsClient used to interact with AWS
 * CloudWatch Logs
 * @param logGroupName  the name of the log group to search within
 * @param logStreamPrefix the prefix of the log streams to search for
 * @param pattern        the pattern to filter log events by
 */
public static void searchLogStreamsAndFilterEvents(CloudWatchLogsClient
cwlClient, String logGroupName, String logStreamPrefix, String pattern) {
    DescribeLogStreamsRequest describeLogStreamsRequest =
    DescribeLogStreamsRequest.builder()
        .logGroupName(logGroupName)
        .logStreamNamePrefix(logStreamPrefix)
        .build();

```

```

        DescribeLogStreamsResponse describeLogStreamsResponse =
        cwlClient.describeLogStreams(describeLogStreamsRequest);
        List<LogStream> logStreams = describeLogStreamsResponse.logStreams();

        for (LogStream logStream : logStreams) {
            String logStreamName = logStream.logStreamName();
            System.out.println("Searching in log stream: " + logStreamName);

            FilterLogEventsRequest filterLogEventsRequest =
            FilterLogEventsRequest.builder()
                .logGroupName(logGroupName)
                .logStreamNames(logStreamName)
                .filterPattern(pattern)
                .build();

            FilterLogEventsResponse filterLogEventsResponse =
            cwlClient.filterLogEvents(filterLogEventsRequest);

            for (FilteredLogEvent event : filterLogEventsResponse.events()) {
                System.out.println(event.message());
            }

            System.out.println("-----"); //
            Separator for better readability
        }
    }
}

```

打印有关指定日志组中最新日志流的元数据。

```

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CloudWatchLogQuery {

```



```

public static void main(final String[] args) {
    final String usage = ""
        Usage:
        <logGroupName>

        Where:
        logGroupName - The name of the log group (for example, /aws/
lambda/ChatAIHandler).
        """;

    if (args.length != 1) {
        System.out.print(usage);
        System.exit(1);
    }

    String logGroupName = "/aws/lambda/ChatAIHandler" ; //args[0];
    CloudWatchLogsClient logsClient = CloudWatchLogsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    describeMostRecentLogStream(logsClient, logGroupName);
}

/**
 * Describes and prints metadata about the most recent log stream in the
 * specified log group.
 *
 * @param logsClient the CloudWatchLogsClient used to interact with AWS
 * CloudWatch Logs
 * @param logGroupName the name of the log group
 */
public static void describeMostRecentLogStream(CloudWatchLogsClient
logsClient, String logGroupName) {
    DescribeLogStreamsRequest streamsRequest =
DescribeLogStreamsRequest.builder()
        .logGroupName(logGroupName)
        .orderBy(OrderBy.LAST_EVENT_TIME)
        .descending(true)
        .limit(1)
        .build();

    try {
        DescribeLogStreamsResponse streamsResponse =
logsClient.describeLogStreams(streamsRequest);
    }
}

```

```
List<LogStream> logStreams = streamsResponse.logStreams();

if (logStreams.isEmpty()) {
    System.out.println("No log streams found for log group: " +
logGroupName);
    return;
}

LogStream stream = logStreams.get(0);
System.out.println("Most Recent Log Stream:");
System.out.println("  Name: " + stream.logStreamName());
System.out.println("  ARN: " + stream.arn());
System.out.println("  Creation Time: " + stream.creationTime());
System.out.println("  First Event Time: " +
stream.firstEventTimestamp());
System.out.println("  Last Event Time: " +
stream.lastEventTimestamp());
System.out.println("  Stored Bytes: " + stream.storedBytes());
System.out.println("  Upload Sequence Token: " +
stream.uploadSequenceToken());

} catch (CloudWatchLogsException e) {
    System.err.println("Failed to describe log stream: " +
e.awsErrorDetails().errorMessage());
}
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [DescribeLogStreams](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeSubscriptionFilters** 和 Amazon SDK 搭配使用

以下代码示例演示如何使用 `DescribeSubscriptionFilters`。

C++

SDK for C++

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

包含所需的文件。

```
#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>
```

列出订阅筛选条件。

```
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();
}

```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API 参考》中的 [DescribeSubscriptionFilters](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

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");
}
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [DescribeSubscriptionFilters](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 [DescribeSubscriptionFilters](#)。

SDK for JavaScript (v2)

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cw1 = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  logGroupName: "GROUP_NAME",
  limit: 5,
};

cw1.describeSubscriptionFilters(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.subscriptionFilters);
  }
});
```

- 有关更多信息，请参阅 [《适用于 JavaScript 的 Amazon SDK 开发人员指南》](#)。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 [DescribeSubscriptionFilters](#)。

Kotlin

SDK for Kotlin

Note

查看 GitHub，了解更多信息。查找完整示例，了解如何在 [Amazon 代码示例存储库](#) 中进行设置和运行。

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
        }
```



```
        limit = 1
    }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { cwlClient
->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name ${filter.filterName} pattern
${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [DescribeSubscriptionFilters](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetLogEvents** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetLogEvents。

CLI

Amazon CLI

以下命令从日志组 my-logs 中名为 20150601 的日志流中检索日志事件：

```
aws logs get-log-events --log-group-name my-logs --log-stream-name 20150601
```

输出：

```
{
  "nextForwardToken":
"f/31961209122447488583055879464742346735121166569214640130",
  "events": [
    {
      "ingestionTime": 1433190494190,
      "timestamp": 1433190184356,
      "message": "Example Event 1"
```

```
    },
    {
      "ingestionTime": 1433190516679,
      "timestamp": 1433190184356,
      "message": "Example Event 1"
    },
    {
      "ingestionTime": 1433190494190,
      "timestamp": 1433190184358,
      "message": "Example Event 2"
    }
  ],
  "nextBackwardToken":
  "b/31961209122358285602261756944988674324553373268216709120"
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 [GetLogEvents](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
  software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import
  software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsResponse;

import java.time.Instant;
import java.time.temporal.ChronoUnit;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class GetLogEvents {

    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <logGroupName> <logStreamName>

            Where:
                logGroupName - The name of the log group (for example,
myloggroup).
                logStreamName - The name of the log stream (for example,
mystream).

            "";

        // if (args.length != 2) {
        //     System.out.print(usage);
        //     System.exit(1);
//     }

        String logGroupName = "WeathertopJavaContainerLogs" ; //args[0];
        String logStreamName = "weathertop-java-stream" ; //args[1];

        Region region = Region.US_EAST_1 ;
        CloudWatchLogsClient cloudWatchLogsClient =
CloudWatchLogsClient.builder()
            .region(region)
            .build();

        getCWLogEvents(cloudWatchLogsClient, logGroupName, logStreamName);
        cloudWatchLogsClient.close();
    }

    public static void getCWLogEvents(CloudWatchLogsClient cloudWatchLogsClient,
```

```

        String logGroupName,
        String logStreamPrefix) {

    try {
        // First, find the exact log stream name
        DescribeLogStreamsRequest describeRequest =
DescribeLogStreamsRequest.builder()
            .logGroupName(logGroupName)
            .logStreamNamePrefix(logStreamPrefix)
            .limit(1) // get the first matching stream
            .build();

        DescribeLogStreamsResponse describeResponse =
cloudWatchLogsClient.describeLogStreams(describeRequest);

        if (describeResponse.getLogStreams().isEmpty()) {
            System.out.println("No matching log streams found for prefix: " +
logStreamPrefix);
            return;
        }

        String exactLogStreamName =
describeResponse.getLogStreams().get(0).getLogStreamName();
        System.out.println("Using exact log stream: " + exactLogStreamName);

        long startTime = Instant.now().minus(7,
ChronoUnit.DAYS).toEpochMilli();
        long endTime = Instant.now().toEpochMilli();

        GetLogEventsRequest getLogEventsRequest =
GetLogEventsRequest.builder()
            .logGroupName(logGroupName)
            .logStreamName(exactLogStreamName) // <-- exact name, not
prefix

            .startTime(startTime)
            .endTime(endTime)
            .startFromHead(true)
            .build();

        GetLogEventsResponse response =
cloudWatchLogsClient.getLogEvents(getLogEventsRequest);

        if (response.getEvents().isEmpty()) {
            System.out.println("No log events found in the past 7 days.");
        } else {

```

```
        response.events().forEach(e -> System.out.println(e.message()));
    }

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [GetLogEvents](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetQueryResults** 和 Amazon SDK 搭配使用

以下代码示例演示如何使用 `GetQueryResults`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [运行大型查询](#)

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
/**
 * Simple wrapper for the GetQueryResultsCommand.
 * @param {string} queryId
 */
_getQueryResults(queryId) {
    return this.client.send(new GetQueryResultsCommand({ queryId }));
}
```

```
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 [GetQueryResults](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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", [])
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [GetQueryResults](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutSubscriptionFilter** 和 Amazon SDK 搭配使用

以下代码示例演示如何使用 PutSubscriptionFilter。

C++

SDK for C++

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

包含所需的文件。

```
#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

创建订阅筛选条件。

```
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;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API 参考》中的 [PutSubscriptionFilter](#)。

Java

适用于 Java 的 SDK 2.x

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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 cw1 = CloudWatchLogsClient.builder()
            .region(region)
            .build();

        putSubFilters(cw1, filter, pattern, logGroup, functionArn);
        cw1.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);
    }
}
```

- 有关 API 的详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [PutSubscriptionFilter](#)。

JavaScript

SDK for JavaScript (v3)

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
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();
```

- 有关 API 的详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 [PutSubscriptionFilter](#)。

SDK for JavaScript (v2)

 Note

查看 [GitHub](#)，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
// 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);
  }
});
```

- 有关更多信息，请参阅 [《适用于 JavaScript 的 Amazon SDK 开发人员指南》](#)。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 [PutSubscriptionFilter](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **StartLiveTail** 和 Amazon SDK 搭配使用

以下代码示例演示如何使用 StartLiveTail。

.NET

适用于 .NET 的 Amazon SDK

包含所需的文件。

```
using Amazon;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
```

启动 Live Tail 会话

```
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;
}
```

您可以通过两种方式处理 Live Tail 会话中的事件：

```
/* 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");
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API 参考》中的 [StartLiveTail](#)。

Go

SDK for Go V2

包含所需的文件。

```
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"
)
```

处理 Live Tail 会话中的事件。

```
func handleEventStreamAsync(stream *cloudwatchlogs.StartLiveTailEventStream) {
    eventsChan := stream.Events()
    for {
        event := <-eventsChan
        switch e := event.(type) {
        case *types.StartLiveTailResponseStreamMemberSessionStart:
            log.Println("Received SessionStart event")
        case *types.StartLiveTailResponseStreamMemberSessionUpdate:
            for _, logEvent := range e.Value.SessionResults {
                log.Println(*logEvent.Message)
            }
        default:
            // Handle on-stream exceptions
            if err := stream.Err(); err != nil {
                log.Fatalf("Error occurred during streaming: %v", err)
            } else if event == nil {
                log.Println("Stream is Closed")
                return
            } else {
                log.Fatalf("Unknown event type: %T", e)
            }
        }
    }
}
```

启动 Live Tail 会话。

```
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)
```

经过一段时间后停止 Live Tail 会话。

```
// Close the stream (which ends the session) after a timeout
time.Sleep(10 * time.Second)
stream.Close()
log.Println("Event stream closed")
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API 参考》中的 [StartLiveTail](#)。

Java

适用于 Java 的 SDK 2.x

包含所需的文件。

```
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;
```

处理 Live Tail 会话中的事件。

```
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();
}

```

启动 Live Tail 会话。

```

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));

```

经过一段时间后停止 Live Tail 会话。

```
/* 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");  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [StartLiveTail](#)。

JavaScript

SDK for JavaScript (v3)

包含所需的文件。

```
import { CloudWatchLogsClient, StartLiveTailCommand } from "@aws-sdk/client-cloudwatch-logs";
```

处理 Live Tail 会话中的事件。

```
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)
}
}
```

启动 Live Tail 会话。

```
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);
}
```

经过一段时间后停止 Live Tail 会话。

```
/* Set a timeout to close the client. This will stop the Live Tail session.
*/
setTimeout(function() {
    console.log("Client timeout");
    client.destroy();
}, 10000);
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 [StartLiveTail](#)。

Kotlin

SDK for Kotlin

包含所需的文件。

```
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
```

启动 Live Tail 会话

```
val client = CloudWatchLogsClient.fromEnvironment()

val request = StartLiveTailRequest {
    logGroupIdentifiers = logGroupIdentifiersVal
    logStreamNames = logStreamNamesVal
    logEventFilterPattern = logEventFilterPatternVal
}

val startTime = System.currentTimeMillis()

try {
    client.startLiveTail(request) { response ->
        val stream = response.responseStream
        if (stream != null) {
            /* Set a timeout to unsubscribe from the flow. This will:
             * 1). Close the stream
             * 2). Stop the Live Tail session
             */
            stream.takeWhile { System.currentTimeMillis() - startTime <
10000 }.collect { value ->
                if (value is StartLiveTailResponseStream.SessionStart) {
                    println(value.asSessionStart())
                } else if (value is
StartLiveTailResponseStream.SessionUpdate) {
                    for (e in value.asSessionUpdate().sessionResults!!) {
                        println(e)
                    }
                } else {
                    throw IllegalArgumentException("Unknown event type")
                }
            }
        }
    }
```

```
        }
    }
    } else {
        throw IllegalArgumentException("No response stream")
    }
}
} catch (e: Exception) {
    println("Exception occurred during StartLiveTail: $e")
    System.exit(1)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [StartLiveTail](#)。

Python

适用于 Python 的 SDK (Boto3)

包含所需的文件。

```
import boto3
import time
from datetime import datetime
```

启动 Live Tail 会话

```
# 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)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [StartLiveTail](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **StartQuery** 和 Amazon SDK 搭配使用

以下代码示例演示如何使用 StartQuery。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [运行大型查询](#)

JavaScript

SDK for JavaScript (v3)

Note

查看 [GitHub](#)，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
/**
 * 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;
  }
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 [StartQuery](#)。

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
def perform_query(self, date_range):
    """
    Performs the actual CloudWatch log query.

    :param date_range: A tuple representing the start and end datetime for
    the query.
    :type date_range: tuple
    :return: A list containing the query results.
    :rtype: list
    """
    client = boto3.client("logs")
    try:
        try:
            start_time = round(
self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(
self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=self.limit,
            )
```

```

        query_id = response["queryId"]
    except client.exceptions.ResourceNotFoundException as e:
        raise DateOutOfBoundsError(f"Resource not found: {e}")
    while True:
        time.sleep(1)
        results = client.get_query_results(queryId=query_id)
        if results["status"] in [
            "Complete",
            "Failed",
            "Cancelled",
            "Timeout",
            "Unknown",
        ]:
            return results.get("results", [])
    except DateOutOfBoundsError:
        return []

def _initiate_query(self, client, date_range, max_logs):
    """
    Initiates the CloudWatch logs query.

    :param date_range: A tuple representing the start and end datetime for
    the query.
    :type date_range: tuple
    :param max_logs: The maximum number of logs to retrieve.
    :type max_logs: int
    :return: The query ID as a string.
    :rtype: str
    """
    try:
        start_time = round(
self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
        )
        end_time = round(
self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
        )
        response = client.start_query(
            logGroupName=self.log_group,
            startTime=start_time,
            endTime=end_time,
            queryString=self.query_string,
            limit=max_logs,

```

```
)  
    return response["queryId"]  
except client.exceptions.ResourceNotFoundException as e:  
    raise DateOutOfBoundsError(f"Resource not found: {e}")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [StartQuery](#)。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon SDK 的 CloudWatch Logs 场景

以下代码示例向您展示了如何通过 Amazon SDK 实施 CloudWatch Logs 中的常见场景。这些场景展示了如何通过 CloudWatch Logs 中调用多个函数或与其他 Amazon Web Services 服务 结合来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- [使用 CloudWatch Logs 运行大型查询](#)
- [使用计划的事件调用 Lambda 函数](#)

使用 CloudWatch Logs 运行大型查询

以下代码示例展示了如何使用 CloudWatch Logs 查询 10,000 条以上的记录。

JavaScript

SDK for JavaScript (v3)

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

这是入口点。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
import { CloudWatchQuery } from "../cloud-watch-query.js";

console.log("Starting a recursive query...");

if (!process.env.QUERY_START_DATE || !process.env.QUERY_END_DATE) {
  throw new Error(
    "QUERY_START_DATE and QUERY_END_DATE environment variables are required.",
  );
}

const cloudWatchQuery = new CloudWatchQuery(new CloudWatchLogsClient({}), {
  logGroupNames: ["/workflows/cloudwatch-logs/large-query"],
  dateRange: [
    new Date(Number.parseInt(process.env.QUERY_START_DATE)),
    new Date(Number.parseInt(process.env.QUERY_END_DATE)),
  ],
});

await cloudWatchQuery.run();

console.log(
  `Queries finished in ${cloudWatchQuery.secondsElapsed} seconds.\nTotal logs found: ${cloudWatchQuery.results.length}`,
);
```

该类可在必要时将查询拆分为多个步骤。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  StartQueryCommand,
  GetQueryResultsCommand,
} from "@aws-sdk/client-cloudwatch-logs";
import { splitDateRange } from "@aws-doc-sdk-examples/lib/utils/util-date.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

class DateOutOfBoundsError extends Error {}
```

```

export class CloudWatchQuery {
  /**
   * Run a query for all CloudWatch Logs within a certain date range.
   * CloudWatch logs return a max of 10,000 results. This class
   * performs a binary search across all of the logs in the provided
   * date range if a query returns the maximum number of results.
   *
   * @param {import('@aws-sdk/client-cloudwatch-logs').CloudWatchLogsClient}
client
   * @param {{ logGroupNames: string[], dateRange: [Date, Date], queryConfig:
{ limit: number } }} config
   */
  constructor(client, { logGroupNames, dateRange, queryConfig }) {
    this.client = client;
    /**
     * All log groups are queried.
     */
    this.logGroupNames = logGroupNames;

    /**
     * The inclusive date range that is queried.
     */
    this.dateRange = dateRange;

    /**
     * CloudWatch Logs never returns more than 10,000 logs.
     */
    this.limit = queryConfig?.limit ?? 10000;

    /**
     * @type {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]}
     */
    this.results = [];
  }

  /**
   * Run the query.
   */
  async run() {
    this.secondsElapsed = 0;
    const start = new Date();
    this.results = await this._largeQuery(this.dateRange);
    const end = new Date();
  }
}

```

```

    this.secondsElapsed = (end - start) / 1000;
    return this.results;
  }

  /**
   * Recursively query for logs.
   * @param {[Date, Date]} dateRange
   * @returns {Promise<import("@aws-sdk/client-cloudwatch-logs").ResultField[
[]>]}
   */
  async _largeQuery(dateRange) {
    const logs = await this._query(dateRange, this.limit);

    console.log(
      `Query date range: ${dateRange
        .map((d) => d.toISOString())
        .join(" to ")}. Found ${logs.length} logs.`
    );

    if (logs.length < this.limit) {
      return logs;
    }

    const lastLogDate = this._getLastLogDate(logs);
    const offsetLastLogDate = new Date(lastLogDate);
    offsetLastLogDate.setMilliseconds(lastLogDate.getMilliseconds() + 1);
    const subDateRange = [offsetLastLogDate, dateRange[1]];
    const [r1, r2] = splitDateRange(subDateRange);
    const results = await Promise.all([
      this._largeQuery(r1),
      this._largeQuery(r2),
    ]);
    return [logs, ...results].flat();
  }

  /**
   * Find the most recent log in a list of logs.
   * @param {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]} logs
   */
  _getLastLogDate(logs) {
    const timestamps = logs
      .map(
        (log) =>
          log.find((fieldMeta) => fieldMeta.field === "@timestamp")?.value,

```

```

    )
    .filter((t) => !!t)
    .map((t) => `${t}Z`)
    .sort();

    if (!timestamps.length) {
        throw new Error("No timestamp found in logs.");
    }

    return new Date(timestamps[timestamps.length - 1]);
}

/**
 * Simple wrapper for the GetQueryResultsCommand.
 * @param {string} queryId
 */
_getQueryResults(queryId) {
    return this.client.send(new GetQueryResultsCommand({ queryId }));
}

/**
 * Starts a query and waits for it to complete.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 */
async _query(dateRange, maxLogs) {
    try {
        const { queryId } = await this._startQuery(dateRange, maxLogs);
        const { results } = await this._waitUntilQueryDone(queryId);
        return results ?? [];
    } catch (err) {
        /**
         * This error is thrown when StartQuery returns an error indicating
         * that the query's start or end date occur before the log group was
         * created.
         */
        if (err instanceof DateOutOfBoundsError) {
            return [];
        }
        throw err;
    }
}

/**

```



```

    * Wrapper for the StartQueryCommand. Uses a static query string
    * for consistency.
    * @param {[Date, Date]} dateRange
    * @param {number} maxLogs
    * @returns {Promise<{ queryId: string }>}
    */
    async _startQuery([startDate, endDate], maxLogs = 10000) {
      try {
        return await this.client.send(
          new StartQueryCommand({
            logGroupNames: this.logGroupNames,
            queryString: "fields @timestamp, @message | sort @timestamp asc",
            startTime: startDate.valueOf(),
            endTime: endDate.valueOf(),
            limit: maxLogs,
          }),
        );
      } catch (err) {
        /** @type {string} */
        const message = err.message;
        if (message.startsWith("Query's end date and time")) {
          // This error indicates that the query's start or end date occur
          // before the log group was created.
          throw new DateOutOfBoundsError(message);
        }

        throw err;
      }
    }

    /**
     * Call GetQueryResultsCommand until the query is done.
     * @param {string} queryId
     */
    _waitUntilQueryDone(queryId) {
      const getResults = async () => {
        const results = await this._getQueryResults(queryId);
        const queryDone = [
          "Complete",
          "Failed",
          "Cancelled",
          "Timeout",
          "Unknown",
        ].includes(results.status);
      };
    }
  }

```

```
    return { queryDone, results };
  };

  return retry(
    { intervalInMs: 1000, maxRetries: 60, quiet: true },
    async () => {
      const { queryDone, results } = await getResults();
      if (!queryDone) {
        throw new Error("Query not done.");
      }

      return results;
    },
  );
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的以下主题。
 - [GetQueryResults](#)
 - [StartQuery](#)

Python

适用于 Python 的 SDK (Boto3)

Note

查看 GitHub，了解更多信息。在 [Amazon 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

此文件调用一个示例模块来管理超过 10,000 个结果的 CloudWatch 查询。

```
import logging
import os
import sys

import boto3
from botocore.config import Config
```

```

from cloudwatch_query import CloudWatchQuery
from date_utilities import DateUtilities

# Configure logging at the module level.
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(filename)s:%(lineno)d - %(message)s",
)

DEFAULT_QUERY_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"

class CloudWatchLogsQueryRunner:
    def __init__(self):
        """
        Initializes the CloudWatchLogsQueryRunner class by setting up date
        utilities
        and creating a CloudWatch Logs client with retry configuration.
        """
        self.date_utilities = DateUtilities()
        self.cloudwatch_logs_client = self.create_cloudwatch_logs_client()

    def create_cloudwatch_logs_client(self):
        """
        Creates and returns a CloudWatch Logs client with a specified retry
        configuration.

        :return: A CloudWatch Logs client instance.
        :rtype: boto3.client
        """
        try:
            return boto3.client("logs", config=Config(retries={"max_attempts":
10}))
        except Exception as e:
            logging.error(f"Failed to create CloudWatch Logs client: {e}")
            sys.exit(1)

    def fetch_environment_variables(self):
        """
        Fetches and validates required environment variables for query start and
        end dates.
        Fetches the environment variable for log group, returning the default
        value if it

```

```

        does not exist.

        :return: Tuple of query start date and end date as integers and the log
group.
        :rtype: tuple
        :raises SystemExit: If required environment variables are missing or
invalid.
        """
        try:
            query_start_date = int(os.environ["QUERY_START_DATE"])
            query_end_date = int(os.environ["QUERY_END_DATE"])
        except KeyError:
            logging.error(
                "Both QUERY_START_DATE and QUERY_END_DATE environment variables
are required."
            )
            sys.exit(1)
        except ValueError as e:
            logging.error(f"Error parsing date environment variables: {e}")
            sys.exit(1)

        try:
            log_group = os.environ["QUERY_LOG_GROUP"]
        except KeyError:
            logging.warning("No QUERY_LOG_GROUP environment variable, using
default value")
            log_group = DEFAULT_QUERY_LOG_GROUP

        return query_start_date, query_end_date, log_group

def convert_dates_to_iso8601(self, start_date, end_date):
    """
    Converts UNIX timestamp dates to ISO 8601 format using DateUtilities.

    :param start_date: The start date in UNIX timestamp.
    :type start_date: int
    :param end_date: The end date in UNIX timestamp.
    :type end_date: int
    :return: Start and end dates in ISO 8601 format.
    :rtype: tuple
    """
    start_date_iso8601 =
self.date_utilities.convert_unix_timestamp_to_iso8601(
    start_date

```

```

    )
    end_date_iso8601 = self.date_utilities.convert_unix_timestamp_to_iso8601(
        end_date
    )
    return start_date_iso8601, end_date_iso8601

def execute_query(
    self,
    start_date_iso8601,
    end_date_iso8601,
    log_group="/workflows/cloudwatch-logs/large-query",
    query="fields @timestamp, @message | sort @timestamp asc"
):
    """
    Creates a CloudWatchQuery instance and executes the query with provided
    date range.

    :param start_date_iso8601: The start date in ISO 8601 format.
    :type start_date_iso8601: str
    :param end_date_iso8601: The end date in ISO 8601 format.
    :type end_date_iso8601: str
    :param log_group: Log group to search: "/workflows/cloudwatch-logs/large-
query"
    :type log_group: str
    :param query: Query string to pass to the CloudWatchQuery instance
    :type query: str
    """
    cloudwatch_query = CloudWatchQuery(
        log_group=log_group,
        query_string=query
    )
    cloudwatch_query.query_logs((start_date_iso8601, end_date_iso8601))
    logging.info("Query executed successfully.")
    logging.info(
        f"Queries completed in {cloudwatch_query.query_duration} seconds.
Total logs found: {len(cloudwatch_query.query_results)}"
    )

def main():
    """
    Main function to start a recursive CloudWatch logs query.
    Fetches required environment variables, converts dates, and executes the
    query.

```

```

"""
logging.info("Starting a recursive CloudWatch logs query...")
runner = CloudWatchLogsQueryRunner()
query_start_date, query_end_date, log_group =
runner.fetch_environment_variables()
start_date_iso8601 = DateUtilities.convert_unix_timestamp_to_iso8601(
    query_start_date
)
end_date_iso8601 =
DateUtilities.convert_unix_timestamp_to_iso8601(query_end_date)
runner.execute_query(start_date_iso8601, end_date_iso8601,
log_group=log_group)

if __name__ == "__main__":
    main()

```

此模块处理超过 10,000 个结果的 CloudWatch 查询。

```

import logging
import time
from datetime import datetime
import threading
import boto3

from date_utilities import DateUtilities

DEFAULT_QUERY = "fields @timestamp, @message | sort @timestamp asc"
DEFAULT_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"

class DateOutOfBoundsError(Exception):
    """Exception raised when the date range for a query is out of bounds."""

    pass

class CloudWatchQuery:
    """
    A class to query AWS CloudWatch logs within a specified date range.

    :vartype date_range: tuple
    :ivar limit: Maximum number of log entries to return.

```

```

        :vartype limit: int
        :log_group str: Name of the log group to query
        :query_string str: query
        """

    def __init__(self, log_group: str = DEFAULT_LOG_GROUP, query_string:
str=DEFAULT_QUERY) -> None:
        self.lock = threading.Lock()
        self.log_group = log_group
        self.query_string = query_string
        self.query_results = []
        self.query_duration = None
        self.datetime_format = "%Y-%m-%d %H:%M:%S.%f"
        self.date_utilities = DateUtilities()
        self.limit = 10000

    def query_logs(self, date_range):
        """
        Executes a CloudWatch logs query for a specified date range and
        calculates the execution time of the query.

        :return: A batch of logs retrieved from the CloudWatch logs query.
        :rtype: list
        """
        start_time = datetime.now()

        start_date, end_date = self.date_utilities.normalize_date_range_format(
            date_range, from_format="unix_timestamp", to_format="datetime"
        )

        logging.info(
            f"Original query:"
            f"\n      START:      {start_date}"
            f"\n      END:        {end_date}"
            f"\n      LOG GROUP: {self.log_group}"
        )
        self.recursive_query((start_date, end_date))
        end_time = datetime.now()
        self.query_duration = (end_time - start_time).total_seconds()

    def recursive_query(self, date_range):
        """
        Processes logs within a given date range, fetching batches of logs
        recursively if necessary.

```

```

        :param date_range: The date range to fetch logs for, specified as a tuple
        (start_timestamp, end_timestamp).
        :type date_range: tuple
        :return: None if the recursive fetching is continued or stops when the
        final batch of logs is processed.
            Although it doesn't explicitly return the query results, this
        method accumulates all fetched logs
            in the `self.query_results` attribute.
        :rtype: None
        """
        batch_of_logs = self.perform_query(date_range)
        # Add the batch to the accumulated logs
        with self.lock:
            self.query_results.extend(batch_of_logs)
        if len(batch_of_logs) == self.limit:
            logging.info(f"Fetched {self.limit}, checking for more...")
            most_recent_log = self.find_most_recent_log(batch_of_logs)
            most_recent_log_timestamp = next(
                item["value"]
                for item in most_recent_log
                if item["field"] == "@timestamp"
            )
            new_range = (most_recent_log_timestamp, date_range[1])
            midpoint = self.date_utilities.find_middle_time(new_range)

            first_half_thread = threading.Thread(
                target=self.recursive_query,
                args=((most_recent_log_timestamp, midpoint),),
            )
            second_half_thread = threading.Thread(
                target=self.recursive_query, args=((midpoint, date_range[1]),)
            )

            first_half_thread.start()
            second_half_thread.start()

            first_half_thread.join()
            second_half_thread.join()

    def find_most_recent_log(self, logs):
        """
        Search a list of log items and return most recent log entry.
        :param logs: A list of logs to analyze.

```



```

        :return: log
        :type :return List containing log item details
        """
        most_recent_log = None
        most_recent_date = "1970-01-01 00:00:00.000"

        for log in logs:
            for item in log:
                if item["field"] == "@timestamp":
                    logging.debug(f"Compared: {item['value']} to
{most_recent_date}")
                    if (
                        self.date_utilities.compare_dates(
                            item["value"], most_recent_date
                        )
                        == item["value"]
                    ):
                        logging.debug(f"New most recent: {item['value']}")
                        most_recent_date = item["value"]
                        most_recent_log = log
            logging.info(f"Most recent log date of batch: {most_recent_date}")
        return most_recent_log

    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(
                    self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(
                    self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )

```

```

        response = client.start_query(
            logGroupName=self.log_group,
            startTime=start_time,
            endTime=end_time,
            queryString=self.query_string,
            limit=self.limit,
        )
        query_id = response["queryId"]
    except client.exceptions.ResourceNotFoundException as e:
        raise DateOutOfBoundsError(f"Resource not found: {e}")
    while True:
        time.sleep(1)
        results = client.get_query_results(queryId=query_id)
        if results["status"] in [
            "Complete",
            "Failed",
            "Cancelled",
            "Timeout",
            "Unknown",
        ]:
            return results.get("results", [])
    except DateOutOfBoundsError:
        return []

def _initiate_query(self, client, date_range, max_logs):
    """
    Initiates the CloudWatch logs query.

    :param date_range: A tuple representing the start and end datetime for
    the query.
    :type date_range: tuple
    :param max_logs: The maximum number of logs to retrieve.
    :type max_logs: int
    :return: The query ID as a string.
    :rtype: str
    """
    try:
        start_time = round(
            self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
        )
        end_time = round(
            self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
        )

```

```
        )
        response = client.start_query(
            logGroupName=self.log_group,
            startTime=start_time,
            endTime=end_time,
            queryString=self.query_string,
            limit=max_logs,
        )
        return response["queryId"]
    except client.exceptions.ResourceNotFoundException as e:
        raise DateOutOfBoundsError(f"Resource not found: {e}")

def _wait_for_query_results(self, client, query_id):
    """
    Waits for the query to complete and retrieves the results.

    :param query_id: The ID of the initiated query.
    :type query_id: str
    :return: A list containing the results of the query.
    :rtype: list
    """
    while True:
        time.sleep(1)
        results = client.get_query_results(queryId=query_id)
        if results["status"] in [
            "Complete",
            "Failed",
            "Cancelled",
            "Timeout",
            "Unknown",
        ]:
            return results.get("results", [])
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。
 - [GetQueryResults](#)
 - [StartQuery](#)

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用计划的事件调用 Lambda 函数

以下代码示例展示如何创建通过 Amazon EventBridge 计划事件调用的 Amazon Lambda 函数。

Java

适用于 Java 的 SDK 2.x

展示如何创建调用 Amazon Lambda 函数的 Amazon EventBridge 计划事件。将 EventBridge 配置为使用 cron 表达式来计划调用 Lambda 函数的时间。在本示例中，您使用 Lambda Java 运行时 API 创建 Lambda 函数。此示例调用不同的 Amazon 服务以执行特定应用场景。此示例展示了如何创建一个应用程序，在其一周年纪念日时向员工发送移动短信表示祝贺。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

适用于 JavaScript 的 SDK (v3)

展示如何创建调用 Amazon Lambda 函数的 Amazon EventBridge 计划事件。将 EventBridge 配置为使用 cron 表达式来计划调用 Lambda 函数的时间。在本示例中，您使用 Lambda JavaScript 运行时 API 创建 Lambda 函数。此示例调用不同的 Amazon 服务以执行特定应用场景。此示例展示了如何创建一个应用程序，在其一周年纪念日时向员工发送移动短信表示祝贺。

有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

该示例也可在 [适用于 JavaScript 的 Amazon SDK v3 开发人员指南](#) 中找到。

本示例中使用的服务

- CloudWatch Logs

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

适用于 Python 的 SDK (Boto3)

此示例说明了如何注册 Amazon Lambda 函数作为计划的 Amazon EventBridge 事件的目标。Lambda 处理程序将友好消息和完整的事件数据写入 Amazon CloudWatch Logs 以供以后检索。

- 部署 Lambda 函数。
- 创建 EventBridge 计划的事件，并将 Lambda 函数作为目标。
- 授予允许 EventBridge 调用 Lambda 函数的权限。
- 打印 CloudWatch Logs 中的最新数据以显示计划调用的结果。
- 清理演示期间创建的所有资源。

此示例最好在 GitHub 上查看。有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 [将 CloudWatch Logs 与 Amazon SDK 结合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon CloudWatch Logs 中的安全性

Amazon 的云安全性的优先级最高。作为 Amazon 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是 Amazon 和您的共同责任。[责任共担模式](#)将其描述为云的安全性和云中的安全性：

- 云安全性 – Amazon 负责保护在 Amazon 云中运行 Amazon 服务的基础结构。Amazon 还向您提供可安全使用的服务。作为[Amazon 合规性计划](#)的一部分，第三方审计人员将定期测试和验证安全性的有效性。要了解适用于 WorkSpaces 的合规性计划，请参阅[合规性计划范围内的 Amazon 服务](#)。
- 云中的安全性 - 您的责任由您使用的 Amazon 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您的公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 Amazon CloudWatch Logs 时应用责任共担模式。它介绍了如何配置 Amazon CloudWatch Logs 以实现您的安全性和合规性目标。您还将了解如何使用其他 Amazon 服务来帮助您的 CloudWatch Logs 资源。

目录

- [Amazon CloudWatch Logs 中的数据保护](#)
- [适用于 Amazon CloudWatch Logs 的 Identity and Access Management](#)
- [Amazon CloudWatch Logs 的合规性验证](#)
- [Amazon CloudWatch Logs 中的恢复能力](#)
- [Amazon CloudWatch Logs 中的基础设施安全性](#)
- [将 CloudWatch Logs 与接口 VPC 终端节点结合使用](#)

Amazon CloudWatch Logs 中的数据保护

Note

除了以下有关 Amazon 中一般数据保护的信息外，CloudWatch Logs 还允许您通过屏蔽来保护日志事件中的敏感数据。有关更多信息，请参阅[通过屏蔽帮助保护敏感的日志数据](#)。

Amazon [责任共担模式](#)适用于 Amazon CloudWatch Logs 中的数据保护。如该模式中所述，Amazon 负责保护运行所有 Amazon Web Services 云的全球基础结构。您负责维护对托管在此基础结构上的内

容的控制。您还负责您所使用的 Amazon Web Services 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅[数据隐私常见问题](#)。

出于数据保护目的，建议您保护 Amazon Web Services 账户 凭证并使用 Amazon IAM Identity Center 或 Amazon Identity and Access Management (IAM) 设置单个用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用 SSL/TLS 与 Amazon 资源进行通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用 Amazon CloudTrail 设置 API 和用户活动日记账记录。有关使用 CloudTrail 跟踪来捕获 Amazon 活动的信息，请参阅《Amazon CloudTrail 用户指南》中的[使用 CloudTrail 跟踪](#)。
- 使用 Amazon 加密解决方案以及 Amazon Web Services 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果在通过命令行界面或 API 访问 Amazon 时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅[《美国联邦信息处理标准 \(FIPS \) 第 140-3 版》](#)。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括使用控制台、API、Amazon CLI 或 Amazon SDK 处理 CloudWatch Logs 或其他 Amazon Web Services 服务。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

静态加密

CloudWatch Logs 使用加密保护静态数据。所有日志组均加密。默认情况下，CloudWatch Logs 服务管理服务器端加密，并使用具有 256 位高级加密标准 Galois/Counter 模式 (AES-GCM) 的服务器端加密来对静态日志数据进行加密。

如果希望管理用于加密和解密日志的密钥，请使用 Amazon KMS 密钥。有关更多信息，请参阅[使用 Amazon Key Management Service 对 CloudWatch Logs 中的日志数据进行加密](#)。

传输中加密

CloudWatch Logs 对传输中的数据使用端到端加密。CloudWatch Logs 服务管理服务器端加密密钥。

适用于 Amazon CloudWatch Logs 的 Identity and Access Management

访问 Amazon CloudWatch Logs 时需要可供 Amazon 用来验证您的请求的凭证。这些凭证必须有权访问 Amazon 资源，例如检索有关您的云资源的 CloudWatch Logs 数据。下面几节提供详细的信息来说明如何使用 [Amazon Identity and Access Management \(IAM\)](#) 和 CloudWatch Logs 控制谁能访问您的资源，从而对这些资源进行保护：

- [身份验证](#)
- [访问控制](#)

身份验证

要提供访问权限，请为您的用户、组或角色添加权限：

- 通过身份提供商在 IAM 中托管的用户：

创建适用于身份联合验证的角色。按照《IAM 用户指南》中[针对第三方身份提供商创建角色 \(联合身份验证 \)](#) 的说明进行操作。

- IAM 用户：
 - 创建您的用户可以担任的角色。按照《IAM 用户指南》中[为 IAM 用户创建角色](#)的说明进行操作。
 - (不推荐使用) 将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中[向用户添加权限 \(控制台 \)](#) 中的说明进行操作。

访问控制

您可以使用有效的凭证来对自己的请求进行身份验证，但您还必须拥有权限才能创建或访问 CloudWatch Logs 资源。例如，您必须拥有创建日志流、创建日志组和执行其他操作的权限。

下面几节介绍如何管理 CloudWatch Logs 的权限。我们建议您先阅读概述。

- [管理您的 CloudWatch Logs 资源的访问权限概述](#)
- [对 CloudWatch Logs 使用基于身份的策略 \(IAM policy \)](#)
- [CloudWatch Logs 权限参考](#)

管理您的 CloudWatch Logs 资源的访问权限概述

要提供访问权限，请为您的用户、组或角色添加权限：

- 通过身份提供商在 IAM 中托管的用户：

创建适用于身份联合验证的角色。按照《IAM 用户指南》中[针对第三方身份提供商创建角色 \(联合身份验证 \)](#) 的说明进行操作。
- IAM 用户：
 - 创建您的用户可以担任的角色。按照《IAM 用户指南》中[为 IAM 用户创建角色](#)的说明进行操作。
 - (不推荐使用) 将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中[向用户添加权限 \(控制台 \)](#) 中的说明进行操作。

主题

- [CloudWatch Logs 资源和操作](#)
- [了解资源所有权](#)
- [管理对资源的访问](#)
- [指定策略元素：操作、效果和主体](#)
- [在策略中指定条件](#)

CloudWatch Logs 资源和操作

在 CloudWatch Logs 中，主资源是日志组、日志流和目标。CloudWatch Logs 不支持子资源 (与主资源一起使用的其他资源)。

这些资源和子资源具有与其关联的唯一 Amazon Resource Name (ARN)，如下表所示。

资源类型	ARN 格式
日志组	使用以下两种方法。以 :* 结尾的第二个是 describe-log-groups CLI 命令和 DescribeLogGroups API 返回的内容。 arn:aws:logs:<i>region</i>:<i>account-id</i> :log-group:<i>log_group_name</i>

资源类型	ARN 格式
	arn:aws:logs:<i>region</i>:<i>account-id</i> :log-group:<i>log_group_name</i> :* 在以下情况下，使用不带尾随字符 :* 的第一个版本： - 在许多 CloudWatch Logs API 的 logGroupIdentifier 输入字段中。 - 在标记 API 的 resourceArn 字段中 - 在 IAM 策略中，当为 TagResource、UntagResource 和 ListTagsForResource 指定权限时。 在 IAM 策略中为所有其他 API 操作指定权限时，使用带有尾随字符 :* 的第二个版本来引用 ARN。
日志流	arn:aws:logs: <i>region</i> : <i>account-id</i> :log-group: <i>log_group_name</i> :log-stream: <i>log_stream_name</i>
目标	arn:aws:logs: <i>region</i> : <i>account-id</i> :destination: <i>destination_name</i>

有关 ARN 的更多信息，请参阅 IAM 用户指南中的 [ARN](#)。有关 CloudWatch 日志 ARN 的更多信息，请参阅 Amazon Web Services 一般参考 中的 [Amazon 资源名称 \(ARN \)](#)。要查看包含 CloudWatch Logs 的策略示例，请参阅 [对 CloudWatch Logs 使用基于身份的策略 \(IAM policy \)](#)。

CloudWatch Logs 提供了一组操作来使用 CloudWatch Logs 资源。有关可用操作的列表，请参阅 [CloudWatch Logs 权限参考](#)。

了解资源所有权

Amazon 账户对在该账户下创建的资源具有所有权，而无论创建资源的人员是谁。具体而言，资源所有者是对资源创建请求进行身份验证的[委托人实体](#)（即根账户、用户或 IAM 角色）的 Amazon 账户。以下示例说明了它的工作原理：

- 如果您使用 Amazon 账户的根账户凭证创建日志组，则您的 Amazon 账户即为该 CloudWatch Logs 资源的所有者。
- 如果您在 Amazon 账户中创建用户并对该用户授予创建 CloudWatch Logs 资源的权限，则该用户也可以创建 CloudWatch Logs 资源。但是，该用户所属的 Amazon 账户拥有 CloudWatch Logs 资源。
- 如果您在您的 Amazon 账户中创建了具有创建 CloudWatch Logs 资源的权限的 IAM 角色，则能够担任该角色的任何人都可以创建 CloudWatch Logs 资源。角色所属的 Amazon 账户拥有该 CloudWatch Logs 资源。

管理对资源的访问

权限策略规定谁可以访问哪些内容。下一节介绍创建权限策略时的可用选项。

Note

本节讨论如何在 CloudWatch Logs 范围内使用 IAM。这里不提供有关 IAM 服务的详细信息。有关完整的 IAM 文档，请参阅 IAM 用户指南中的[什么是 IAM？](#)。有关 IAM policy 语法和说明的信息，请参阅 IAM 用户指南中的[IAM policy 参考](#)。

附加到 IAM 身份的策略称作基于身份的策略（IAM policy），附加到资源的策略称作基于资源的策略。CloudWatch Logs 支持适用于目标的基于身份的策略以及基于资源的策略，您可以使用这些策略来启用跨账户订阅。有关更多信息，请参阅[跨账户跨区域订阅](#)。

主题

- [日志组权限和 Contributor Insights](#)
- [基于资源的策略](#)

日志组权限和 Contributor Insights

Contributor Insights 是 CloudWatch 的一项功能，使用此功能，您能够分析日志组中的数据并创建显示贡献者数据的时间序列。您可以查看有关前 N 个贡献者、独特贡献者总数及其使用情况的指标。有关更多信息，请参阅[使用 Contributor Insights 分析高基数数据](#)。

当您向用户授予 `cloudwatch:PutInsightRule` 和 `cloudwatch:GetInsightRuleReport` 权限，该用户可以创建一个规则来评估 CloudWatch Logs 中的任何日志组，然后查看结果。结果可以包含这些日志组的贡献者数据。确保仅将这些权限授予能够查看此数据的用户。

基于资源的策略

CloudWatch Logs 支持适用于目标的基于资源的策略，您可以使用该策略启用跨账户订阅。有关更多信息，请参阅 [步骤 1：创建目标](#)。可使用 [PutDestination](#) API 创建目标，您可使用 [PutDestinationPolicy](#) API 向目标添加资源策略。在以下示例中，允许账户 ID 为 111122223333 的另一个 Amazon 账户将其日志组订阅到目标 `arn:aws:logs:us-east-1:123456789012:destination:testDestination`。

JSON

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111122223333"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-east-1:123456789012:destination:testDestination"
    }
  ]
}
```

指定策略元素：操作、效果和主体

对于每种 CloudWatch Logs 资源，该服务都定义了一组 API 操作。为授予执行这些 API 操作的权限，CloudWatch Logs 定义了一组您可以在策略中指定的操作。某些 API 操作可能需要多个操作的

权限才能执行 API 操作。有关资源和 API 操作的更多信息，请参阅 [CloudWatch Logs 资源和操作](#) 和 [CloudWatch Logs 权限参考](#)。

以下是基本的策略元素：

- 资源 – 您使用 Amazon Resource Name (ARN) 来标识策略应用到的资源。有关更多信息，请参阅 [CloudWatch Logs 资源和操作](#)。
- 操作 – 您可以使用操作关键字标识要允许或拒绝的资源操作。例如，`logs.DescribeLogGroups` 权限允许执行 `DescribeLogGroups` 操作的用户权限。
- 效果 – 用于指定用户请求特定操作时的效果（可以是允许或拒绝）。如果没有显式授予（允许）对资源的访问权限，则隐式拒绝访问。您也可显式拒绝对资源的访问，这样可确保用户无法访问该资源，即使有其他策略授予了访问权限的情况下也是如此。
- 主体 – 在基于身份的策略（IAM 策略）中，附加了策略的用户是隐式主体。对于基于资源的策略，您可以指定要接收权限的用户、账户、服务或其他实体（仅适用于基于资源的策略）。CloudWatch Logs 支持适用于目标的基于资源的策略。

有关 IAM policy 语法和描述的更多信息，请参阅 IAM 用户指南中的 [Amazon IAM policy 参考](#)。

有关显示所有 CloudWatch Logs API 操作及其适用资源的表，请参阅 [CloudWatch Logs 权限参考](#)。

在策略中指定条件

当您授予权限时，可使用访问策略语言来指定规定策略何时生效的条件。例如，您可能希望策略仅在特定日期后应用。有关使用策略语言指定条件的更多信息，请参阅《IAM 用户指南》中的 [条件](#)。

要表示条件，您可以使用预定义的条件键。有关每个 Amazon 服务支持的上下文键列表以及 Amazon 范围的策略键列表，请参阅 [Amazon 服务的操作、资源和条件键](#) 以及 [Amazon 全局条件上下文键](#)。

Note

您可以使用标签控制对 CloudWatch Logs 资源（包括日志组和目标）的访问。由于日志组和日志流之间存在分层关系，因此在日志组级别控制对日志流的访问。有关使用标签控制访问的更多信息，请参阅 [使用标签控制对亚马逊云科技资源的访问](#)。

对 CloudWatch Logs 使用基于身份的策略（IAM policy）

本主题提供了基于身份的策略的示例，在这些策略中，账户管理员可以向 IAM 身份（即：用户、组和角色）附加权限策略。

⚠ Important

我们建议您首先阅读以下介绍性主题，这些主题讲解了管理 CloudWatch Logs 资源访问的基本概念和选项。有关更多信息，请参阅 [管理您的 CloudWatch Logs 资源的访问权限概述](#)。

本主题包含以下内容：

- [使用 CloudWatch 控制台所需的权限](#)
- [Amazon 用于 CloudWatch Logs 的托管（预定义）策略](#)
- [客户管理型策略示例](#)

下面是权限策略的示例：

JSON

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

本策略具有一个语句，该语句授予了创建日志组和日志流、将事件上传到日志流和列出有关日志流的详细信息的权限。

Resource 值结尾的通配符 (*) 表示一个语句，该语句授予了对任何日志组执行 logs:CreateLogGroup、logs:CreateLogStream、logs:PutLogEvents 和 logs:DescribeLogStreams 操作的权限。要将此权限限制到特定日志组，请使用将资源 ARN 中的通配符 (*) 替换为特定日志组 ARN。有关 IAM policy 语句中各部分的更多信息，请参阅 IAM 用户指南中的 [IAM policy 元素引用](#)。有关显示所有 CloudWatch Logs 操作的列表，请参阅 [CloudWatch Logs 权限参考](#)。

使用 CloudWatch 控制台所需的权限

用户若要能够在 CloudWatch 控制台中使用 CloudWatch Logs，则必须拥有一组最低的权限来允许其描述自己 Amazon 账户中的其他 Amazon 资源。要在 CloudWatch 控制台中使用 CloudWatch Logs，您必须拥有来自以下服务的权限：

- CloudWatch
- CloudWatch Logs
- OpenSearch Service
- IAM
- Kinesis
- Lambda
- Amazon S3

如果创建比必需的最低权限更为严格的 IAM policy，对于附加了该 IAM policy 的用户，控制台将无法按预期正常运行。要确保这些用户仍可使用 CloudWatch 控制台，也可向用户附加 CloudWatchReadOnlyAccess 托管策略，如 [Amazon 用于 CloudWatch Logs 的托管（预定义）策略](#) 中所述。

对于只需要调用 Amazon CLI 或 CloudWatch Logs API 的用户，无需为其提供最低控制台权限。

对于未使用控制台管理日志订阅的用户，使用 CloudWatch 控制台所需的完整权限集为：

- cloudwatch:GetMetricData
- cloudwatch:ListMetrics
- logs:CancelExportTask
- logs:CreateExportTask
- logs:CreateLogGroup
- logs:CreateLogStream

- logs:DeleteLogGroup
- logs:DeleteLogStream
- logs:DeleteMetricFilter
- logs:DeleteQueryDefinition
- logs:DeleteRetentionPolicy
- logs:DeleteSubscriptionFilter
- logs:DescribeExportTasks
- logs:DescribeLogGroups
- logs:DescribeLogStreams
- logs:DescribeMetricFilters
- logs:DescribeQueryDefinitions
- logs:DescribeQueries
- logs:DescribeSubscriptionFilters
- logs:FilterLogEvents
- logs:GetLogEvents
- logs:GetLogGroupFields
- logs:GetLogRecord
- logs:GetQueryResults
- logs:PutMetricFilter
- logs:PutQueryDefinition
- logs:PutRetentionPolicy
- logs:StartQuery
- logs:StopQuery
- logs:PutSubscriptionFilter
- logs:TestMetricFilter

对于同时使用控制台来管理日志订阅的用户，还需要以下权限：

- es:DescribeElasticsearchDomain
- es:ListDomainNames

- iam:AttachRolePolicy
- iam:CreateRole
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetRole
- iam:ListAttachedRolePolicies
- iam:ListRoles
- kinesis:DescribeStreams
- kinesis:ListStreams
- lambda:AddPermission
- lambda:CreateFunction
- lambda:GetFunctionConfiguration
- lambda:ListAliases
- lambda:ListFunctions
- lambda:ListVersionsByFunction
- lambda:RemovePermission
- s3:ListBuckets

Amazon用于 CloudWatch Logs 的托管（预定义）策略

Amazon通过提供由Amazon创建和管理的独立 IAM policy 来满足许多常用案例的要求。托管策略可针对常见使用案例授予必要权限，因此，您无需自行调查具体需要哪些权限。有关更多信息，请参阅《IAM 用户指南》中的 [Amazon 托管式策略](#)。

以下 Amazon 托管式策略（可附加到您账户中的用户和角色）特定于 CloudWatch Logs：

- CloudWatchLogsFullAccess – 授予对 CloudWatch Logs 的完全访问权限。
- CloudWatchLogsReadOnlyAccess – 授予对 CloudWatch Logs 的只读访问权限。

CloudWatchLogsFullAccess

CloudWatchLogsFullAccess 策略授予对 CloudWatch Logs 的完全访问权限。该策略包含 `cloudwatch:GenerateQuery` 和 `cloudwatch:GenerateQueryResultsSummary` 权限，以便

具有此策略的用户可以从自然语言提示生成 [CloudWatch Logs Insights](#) 查询字符串。要查看该策略的全部内容，请参阅《Amazon 托管式策略参考指南》中的 [CloudWatchLogsFullAccess](#)。

CloudWatchLogsReadOnlyAccess

CloudWatchLogsReadOnlyAccess 策略授予对 CloudWatch Logs 的只读访问权限。它包含 `cloudwatch:GenerateQuery` 和 `cloudwatch:GenerateQueryResultsSummary` 权限，以便具有此策略的用户可以从自然语言提示生成 [CloudWatch Logs Insights](#) 查询字符串。要查看该策略的全部内容，请参阅《Amazon 托管式策略参考指南》中的 [CloudWatchLogsReadOnlyAccess](#)。

CloudWatchOpenSearchDashboardsFullAccess

CloudWatchOpenSearchDashboardsFullAccess 策略授予创建、管理和删除与 OpenSearch Service 的集成以及在這些集成中创建、删除和管理公开发布的日志控制面板的权限。有关更多信息，请参阅 [使用 Amazon OpenSearch Service 进行分析](#)。

要查看该策略的全部内容，请参阅《Amazon 托管式策略参考指南》中的 [CloudWatchOpenSearchDashboardsFullAccess](#)。

CloudWatchOpenSearchDashboardAccess

CloudWatchOpenSearchDashboardAccess 策略授予查看使用 Amazon OpenSearch Service 分析创建的公开发布的日志控制面板的权限。有关更多信息，请参阅 [使用 Amazon OpenSearch Service 进行分析](#)。

Important

除了授予此策略之外，要使角色或用户能够查看公开发布的日志控制面板，您还必须在创建与 OpenSearch Service 的集成时指定它们。有关更多信息，请参阅 [步骤 1：创建与 OpenSearch Service 的集成](#)。

要查看该策略的全部内容，请参阅《Amazon 托管式策略参考指南》中的 [CloudWatchOpenSearchDashboardAccess](#)。

CloudWatchLogsCrossAccountSharingConfiguration

CloudWatchLogsCrossAccountSharingConfiguration 策略授予可创建、管理和查看可观测性访问管理器链接的权限，用于在账户之间共享 CloudWatch 日志资源。有关更多信息，请参阅 [CloudWatch 跨账户可观察性](#)。

要查看该策略的全部内容，请参阅《Amazon 托管策略参考指南》中的 [CloudWatchLogsCrossAccountSharingConfiguration](#)。

CloudWatch Logs 对Amazon托管策略的更新

查看有关 CloudWatch Logs 的Amazon托管策略更新的详细信息（从该服务开始跟踪这些更改开始）。有关此页面更改的自动提示，请订阅 CloudWatch Logs 文档历史记录页面上的 RSS 源。

更改	描述	日期
CloudWatchLogsFullAccess – 更新现有策略。	CloudWatch Logs 已向 CloudWatchLogsFullAccess 添加了权限。 添加了 cloudwatc h:GenerateQueryRes ultsSummary 的权限，以 允许生成查询结果的自然语言 摘要。	2025 年 5 月 20 日
CloudWatchLogsRead OnlyAccess - 更新现有策略。	CloudWatch Logs 已向 CloudWatchLogsRead OnlyAccess 添加了权限。 添加了 cloudwatc h:GenerateQueryRes ultsSummary 的权限，以 允许生成查询结果的自然语言 摘要。	2025 年 5 月 20 日
CloudWatchLogsFullAccess – 更新现有策略。	CloudWatch Logs 已向 CloudWatchLogsFullAccess 添加了权限。 添加了 Amazon OpenSearc h Service 和 IAM 的权限，	2024 年 12 月 1 日

更改	描述	日期
	以便将 CloudWatch Logs 与 OpenSearch Service 集成，从而实现某些功能。	
CloudWatchOpenSearchDashboardsFullAccess – 新的 IAM 策略。	CloudWatch Logs 新增了 IAM 策略 CloudWatchOpenSearchDashboardsFullAccess。此策略授予创建、管理和删除与 OpenSearch Service 的集成以及在这些集成中创建、管理和删除公开发布的日志控制面板的权限。有关更多信息，请参阅 使用 Amazon OpenSearch Service 进行分析 。	2024 年 12 月 1 日
CloudWatchOpenSearchDashboardAccess – 新的 IAM 策略。	CloudWatch Logs 新增了 IAM 策略 CloudWatchOpenSearchDashboardAccess。此策略授予查看由 Amazon OpenSearch Service 提供支持的公开发布的日志控制面板的权限。有关更多信息，请参阅 使用 Amazon OpenSearch Service 进行分析 。	2024 年 12 月 1 日

更改	描述	日期
CloudWatchLogsFullAccess – 更新现有策略。	CloudWatch Logs 已向 CloudWatchLogsFullAccess 添加了权限。 添加了 <code>cloudwatch:GenerateQuery</code> 权限，以便拥有此策略的用户可以根据自然语言提示来生成 CloudWatch Logs Insights 查询字符串。	2023 年 11 月 27 日
CloudWatchLogsReadOnlyAccess - 更新现有策略。	CloudWatch 已向 CloudWatchLogsReadOnlyAccess 添加了权限。 添加了 <code>cloudwatch:GenerateQuery</code> 权限，以便拥有此策略的用户可以根据自然语言提示来生成 CloudWatch Logs Insights 查询字符串。	2023 年 11 月 27 日
CloudWatchLogsReadOnlyAccess - 更新到现有策略	CloudWatch Logs 已向 CloudWatchLogsReadOnlyAccess 添加了权限。 添加了 <code>logs:StartLiveTail</code> 和 <code>logs:StopLiveTail</code> 权限，以便使用此策略的用户可以使用控制台启动和停止 CloudWatch Logs live tail 会话。有关更多信息，请参阅 使用 Live Tail 近乎实时地查看日志。	2023 年 6 月 6 日

更改	描述	日期
CloudWatchLogsCrossAccountSharingConfiguration – 新策略	Amazon CloudWatch Logs 添加了新策略，助力您管理用于分享 Amazon CloudWatch Logs 指标的 CloudWatch 跨账户可观测性链接。 有关更多信息，请参阅 CloudWatch 跨账户可观测性。	2022 年 11 月 27 日
CloudWatchLogsReadOnlyAccess - 更新到现有策略	CloudWatch Logs 已向 CloudWatchLogsReadOnlyAccess 添加了权限。 添加了 <code>oam:ListSinks</code> 和 <code>oam:ListAttachedLinks</code> 权限，以便使用此策略的用户可以借助控制台在 CloudWatch 跨账户可观测性中查看源账户共享的数据。	2022 年 11 月 27 日

客户管理型策略示例

您可以创建您自己的自定义 IAM policy，以授予对 CloudWatch Logs 操作和资源的相关权限。您可以将这些自定义策略附加到需要这些权限的用户或组。

本节的用户策略示例介绍如何授予对各 CloudWatch Logs 操作的权限。当您使用 CloudWatch Logs API、Amazon 开发工具包或 Amazon CLI 时，可以使用这些策略。

示例

- [示例 1：允许对 CloudWatch Logs 进行完全访问](#)
- [示例 2：允许对 CloudWatch Logs 进行只读访问](#)
- [示例 3：允许访问一个日志组](#)

示例 1：允许对 CloudWatch Logs 进行完全访问

以下策略允许用户访问所有 CloudWatch Logs 操作。

JSON

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

示例 2：允许对 CloudWatch Logs 进行只读访问

Amazon 提供了一个 CloudWatchLogsReadOnlyAccess 策略，该策略允许对 CloudWatch Logs 数据进行只读访问。该策略包含以下权限。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:Describe*",
        "logs:Get*",
        "logs:List*",
        "logs:StartQuery",
        "logs:StopQuery",
        "logs:TestMetricFilter",
        "logs:FilterLogEvents",
        "logs:StartLiveTail",
        "logs:StopLiveTail",

```

```
        "cloudwatch:GenerateQuery"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

示例 3：允许访问一个日志组

以下策略允许用户在一个指定的日志组中读取和写入日志事件。

Important

Resource 行中日志组名称末尾的 `:*` 是必需的，以指示该策略适用于此日志组中的所有日志流。如果省略 `:*`，则不会强制执行该策略。

JSON

```
{
  "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:*"
    }
  ]
}
```


使用标记和 IAM policy 在日志组级别进行控制

您可以为用户授予某些日志组的访问权限，同时禁止他们访问其他日志组。为此，请标记您的日志组，并使用引用这些标记的 IAM policy。要将标签应用于日志组，您需要拥有 `logs:TagResource` 或 `logs:TagLogGroup` 权限。这既适用于在创建日志组时为其分配标签，也适用于稍后分配标签。

有关标记日志组的更多信息，请参阅 [在 Amazon CloudWatch Logs 中标记日志组](#)。

在标记日志组时，您可以为用户授予 IAM policy 以仅允许访问具有特定标记的日志组。例如，以下策略语句仅授予 Green 标签键值为 Team 的日志组的访问权限。

JSON

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

StopQuery 和 StopLiveTail API 操作不与传统意义上的 Amazon 资源交互。它们不会返回任何数据、放置任何数据或以任何方式修改资源。相反，它们仅对给定的 Live Tail 会话或给定的 CloudWatch Logs Insights 查询进行操作，这些会话和查询不属于资源。因此，在 IAM policy 中为这些操作指定 Resource 字段时，必须将 Resource 字段的值设置为 *，如下例所示。

JSON

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

```
    "Statement":
      [ {
        "Effect": "Allow",
        "Action": [
          "logs:StopQuery",
          "logs:StopLiveTail"
        ],
        "Resource": "*"
      }
    ]
  }
```

有关使用 IAM policy 语句的更多信息，请参阅 IAM 用户指南中的[使用策略控制访问](#)。

CloudWatch Logs 权限参考

在设置 [访问控制](#) 和编写您可挂载到 IAM 身份的权限策略（基于身份的策略）时，可以使用下表作为参考。此表列出每个 CloudWatch Logs API 操作以及您可授权执行该操作的相应操作。请在策略的 Action 字段中指定这些操作。对于 Resource 字段，您可以指定日志组或日志流的 ARN，或者指定 * 来表示所有 CloudWatch Logs 资源。

您可以在 CloudWatch Logs 策略中使用 Amazon 范围的条件键来表达条件。有关 Amazon 范围的键的完整列表，请参阅 IAM 用户指南中的[Amazon 全局和 IAM 条件上下文键](#)。

 Note

要指定操作，请在 API 操作名称之前使用 logs: 前缀。例如：logs:CreateLogGroup、logs:CreateLogStream 或 logs:*（适用于所有 CloudWatch Logs 操作）。

CloudWatch Logs API 操作和必需的操作权限

CloudWatch Logs API 操作	所需权限（API 操作）
CancelExportTask	logs:CancelExportTask 需要取消待处理或正在运行的导出任务。

CloudWatch Logs API 操作	所需权限 (API 操作)
CreateExportTask	<code>logs:CreateExportTask</code> 将数据从日志组导出到 Amazon S3 存储桶所需。
CreateLogGroup	<code>logs:CreateLogGroup</code> 需要创建新的日志组。
CreateLogStream	<code>logs:CreateLogStream</code> 需要在日志组中创建新的日志流。
DeleteDestination	<code>logs:DeleteDestination</code> 需要删除日志目标并对其禁用所有订阅筛选器。
DeleteLogGroup	<code>logs:DeleteLogGroup</code> 需要删除日志组和所有关联的存档日志事件。
DeleteLogStream	<code>logs:DeleteLogStream</code> 需要删除日志流和所有关联的存档日志事件。
DeleteMetricFilter	<code>logs:DeleteMetricFilter</code> 需要删除与日志组关联的指标筛选器。
DeleteQueryDefinition	<code>logs:DeleteQueryDefinition</code> 删除 CloudWatch Logs Insights 中保存的查询定义所需。

CloudWatch Logs API 操作	所需权限 (API 操作)
DeleteResourcePolicy	<code>logs:DeleteResourcePolicy</code> 删除 CloudWatch Logs 资源策略所需。
DeleteRetentionPolicy	<code>logs:DeleteRetentionPolicy</code> 需要删除日志组的保留策略。
DeleteSubscriptionFilter	<code>logs:DeleteSubscriptionFilter</code> 需要删除与日志组关联的订阅筛选器。
DescribeDestinations	<code>logs:DescribeDestinations</code> 需要查看与账户关联的所有目标。
DescribeExportTasks	<code>logs:DescribeExportTasks</code> 需要查看与账户关联的所有导出任务。
DescribeLogGroups	<code>logs:DescribeLogGroups</code> 需要查看与账户关联的所有日志组。
DescribeLogStreams	<code>logs:DescribeLogStreams</code> 需要查看与日志组关联的所有日志流。
DescribeMetricFilters	<code>logs:DescribeMetricFilters</code> 需要查看与日志组关联的所有指标。

CloudWatch Logs API 操作	所需权限 (API 操作)
DescribeQueryDefinitions	<code>logs:DescribeQueryDefinitions</code> 查看 CloudWatch Logs Insights 中保存的查询定义列表所需。
DescribeQueries	<code>logs:DescribeQueries</code> 查看已计划、正在执行或最近已执行的 CloudWatch Logs Insights 查询列表所需。
DescribeResourcePolicies	<code>logs:DescribeResourcePolicies</code> 查看 CloudWatch Logs 资源策略列表所需。
DescribeSubscriptionFilters	<code>logs:DescribeSubscriptionFilters</code> 需要查看与日志组关联的所有订阅筛选器。
FilterLogEvents	<code>logs:FilterLogEvents</code> 需要按照日志组筛选器模式对日志事件进行排序。
GetLogEvents	<code>logs:GetLogEvents</code> 需要从日志流中检索日志事件。
GetLogGroupFields	<code>logs:GetLogGroupFields</code> 需要检索日志组中日志事件内包含的字段列表。

CloudWatch Logs API 操作	所需权限 (API 操作)
GetLogRecord	<code>logs:GetLogRecord</code> 需要从单个日志事件中检索详细信息。
GetLogObject	<code>logs:GetLogRecord</code> 获取通过 PutOpenTelemetryLogs API 摄取的大部分日志事件的内容所需的权限。
GetQueryResults	<code>logs:GetQueryResults</code> 检索 CloudWatch Logs Insights 查询的结果所需。
ListEntitiesForLogGroup (CloudWatch 控制台专用权限)	<code>logs:ListEntitiesForLogGroup</code> 查找与日志组关联的实体所需的权限。在 CloudWatch 控制台中探索相关日志所需的权限。
ListLogGroupsForEntity (CloudWatch 控制台专用权限)	<code>logs:ListLogGroupsForEntity</code> 查找与实体关联的日志组所需的权限。在 CloudWatch 控制台中探索相关日志所需的权限。
ListTagsLogGroup	<code>logs:ListTagsLogGroup</code> 需要列出与日志组关联的标签。
ListLogGroups	<code>logs:DescribeLogGroups</code> 需要查看与账户关联的所有日志组。

CloudWatch Logs API 操作	所需权限 (API 操作)
PutDestination	<code>logs:PutDestination</code> 创建或更新目标日志流 (例如 , Kinesis 流) 所需。
PutDestinationPolicy	<code>logs:PutDestinationPolicy</code> 需要创建或更新与现有日志目标关联的访问策略。
PutLogEvents	<code>logs:PutLogEvents</code> 需要将一批日志事件上传到日志流。
PutMetricFilter	<code>logs:PutMetricFilter</code> 需要创建或更新指标筛选器并将其与日志组关联。
PutQueryDefinition	<code>logs:PutQueryDefinition</code> 在 CloudWatch Logs Insights 中保存查询所需。
PutResourcePolicy	<code>logs:PutResourcePolicy</code> 创建 CloudWatch Logs 资源策略所需。
PutRetentionPolicy	<code>logs:PutRetentionPolicy</code> 需要设置日志组中的日志事件的保存 (保留) 天数。

CloudWatch Logs API 操作	所需权限 (API 操作)
PutSubscriptionFilter	<code>logs:PutSubscriptionFilter</code> 需要创建或更新订阅筛选器并将其与日志组关联。
StartQuery	<code>logs:StartQuery</code> 启动 CloudWatch Logs Insights 查询所需。
StopQuery	<code>logs:StopQuery</code> 停止正在执行的 CloudWatch Logs Insights 查询所需。
TagLogGroup	<code>logs:TagLogGroup</code> 需要添加或更新日志组标签。
TestMetricFilter	<code>logs:TestMetricFilter</code> 需要针对日志事件消息采样测试筛选器模式。

为 CloudWatch Logs 使用服务相关角色

Amazon CloudWatch Logs 使用 Amazon Identity and Access Management (IAM) [服务相关角色](#)。服务相关角色是一种独特类型的 IAM 角色，它与 CloudWatch Logs 直接相关。服务相关角色由 CloudWatch Logs 预定义，并包含该服务代表您调用其他 Amazon 服务所需的一切权限。

您可以使用服务相关角色轻松设置 CloudWatch Logs，因为您不必手动添加所需的权限。CloudWatch Logs 定义其服务相关角色的权限，除非另外定义，否则只有 CloudWatch Logs 可以代入该角色。定义的权限包括信任策略和权限策略。这些权限策略不能附加到任何其他 IAM 实体。

有关支持服务相关角色的其他服务的信息，请参阅[与 IAM 配合使用的 Amazon 服务](#)。查找在 Service-Linked Role (服务相关角色) 列中具有 Yes (是) 值的服务。选择 Yes (是) 与查看该服务的服务相关角色文档的链接。

CloudWatch Logs 的服务相关角色权限

CloudWatch Logs 使用名为 `AWSServiceRoleForLogDelivery` 的服务相关角色。CloudWatch Logs 使用此服务相关角色将日志直接写入 Firehose。有关更多信息，请参阅[启用从 Amazon 服务进行日志记录](#)。

`AWSServiceRoleForLogDelivery` 服务相关角色信任以下服务代入该角色：

- `logs.amazonaws.com`

角色权限策略允许 CloudWatch Logs 对指定资源完成以下操作：

- 操作：所有 Firehose 流上标记有 `LogDeliveryEnabled` 键且值为 `True` 的 `firehose:PutRecord` 和 `firehose:PutRecordBatch`。当您创建订阅以将日志传送到 Firehose 时，此标记会自动附加到 Firehose 流。

您必须配置权限以允许 IAM 实体创建、编辑或删除服务相关角色。此实体可以是用户、组或角色。有关更多信息，请参阅 IAM 用户指南中的[服务相关角色权限](#)。

为 CloudWatch Logs 创建服务相关角色

您无需手动创建服务相关角色。当您为日志设置直接发送到 Amazon Web Services 管理控制台、Amazon CLI 或 Amazon API 中的 Firehose 流时，CloudWatch Logs 将为您创建服务相关角色。

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。当您再次将日志设置为直接发送到 Firehose 流时，CloudWatch Logs 将再次为您创建服务相关角色。

为 CloudWatch Logs 编辑服务相关角色

创建角色后，CloudWatch Logs 不允许您编辑 `AWSServiceRoleForLogDelivery` 或任何其他与服务相关的角色。由于多个实体可能引用该角色，因此无法更改角色的名称。不过，您可以使用 IAM 编辑角色的说明。有关更多信息，请参阅《IAM 用户指南》中的[编辑服务相关角色](#)。

为 CloudWatch Logs 删除服务相关角色

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，必须先清除服务相关角色的资源，然后才能手动删除它。

 Note

如果在您试图删除资源时 CloudWatch Logs 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，请等待几分钟后重试。

删除 AWSServiceRoleForLogDelivery 服务相关角色所使用的 CloudWatch Logs 资源

- 停止将日志直接发送到 Firehose 流。

使用 IAM 手动删除 服务相关角色

使用 IAM 控制台、Amazon CLI 或 Amazon API 来删除 AWSServiceRoleForLogDelivery 服务相关角色。有关更多信息，请参阅[删除服务相关角色](#)

CloudWatch Logs 服务相关角色支持的区域

CloudWatch Logs 支持在服务可用的所有Amazon区域中使用服务相关角色。有关更多信息，请参阅[CloudWatch Logs 区域和终端节点](#)。

Amazon 服务相关角色的 CloudWatch Logs 更新

查看自此服务开始跟踪这些更改以来有关 CloudWatch Logs 的 Amazon 服务相关角色更新的详细信息。有关此页面更改的自动提示，请订阅 CloudWatch Logs 文档历史记录页面上的 RSS 源。

更改	描述	日期
AWSServiceRoleForLogDelivery 服务相关角色策略 – 对现有策略的更新	CloudWatch Logs 更改了与 AWSServiceRoleForLogDelivery 服务相关角色关联的 IAM policy 权限。更改内容如下： • <code>firehose:ResourceTag/LogDeliveryEnabled</code>: "true" 条件键已更改为 <code>aws:Resou</code>	2021 年 7 月 15 日

更改	描述	日期
	rcceTag/LogDelivery Enabled": "true" 。	
CloudWatch Logs 开始跟踪更改	CloudWatch Logs 开始跟踪其 Amazon 托管策略的更改。	2021 年 6 月 10 日

Amazon CloudWatch Logs 的合规性验证

要了解某个 Amazon Web Services 服务是否在特定合规性计划范围内，请参阅[合规性计划范围内的 Amazon Web Services 服务](#)，然后选择您感兴趣的合规性计划。有关常规信息，请参阅[Amazon Web Services 合规性计划](#)。

您可以使用 Amazon Artifact 下载第三方审计报告。有关更多信息，请参阅[在 Amazon Artifact 中下载报告](#)。

您在使用 Amazon Web Services 服务时的合规性责任由您的数据的敏感性、您公司的合规性目标以及适用的法律法规决定。有关您在使用 Amazon Web Services 服务时的合规责任的更多信息，请参阅[Amazon 安全性文档](#)。

Amazon CloudWatch Logs 中的恢复能力

Amazon 全球基础设施围绕 Amazon 区域和可用区构建。区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错能力和可扩展性。

有关 Amazon 区域和可用区的更多信息，请参阅[Amazon 全球基础设施](#)。

Amazon CloudWatch Logs 中的基础设施安全性

作为一项托管式服务，Amazon CloudWatch Logs 受 Amazon 全球网络安全保护。有关 Amazon 安全服务以及 Amazon 如何保护基础设施的信息，请参阅[Amazon 云安全性](#)。要按照基础设施安全最佳实

设计您的 Amazon 环境，请参阅《安全性支柱 Amazon Well-Architected Framework》中的[基础设施保护](#)。

您可以使用 Amazon 发布的 API 调用并通过网络访问 CloudWatch Logs。客户端必须支持以下内容：

- 传输层安全性协议 (TLS)。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密 (PFS) 的密码套件，例如 DHE (临时 Diffie-Hellman) 或 ECDHE (临时椭圆曲线 Diffie-Hellman)。大多数现代系统 (如 Java 7 及更高版本) 都支持这些模式。

将 CloudWatch Logs 与接口 VPC 终端节点结合使用

如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 托管 Amazon 资源，则可以在您的 VPC 和 CloudWatch Logs 之间建立私有连接。您可以使用此连接将日志发送到 CloudWatch Logs，而无需通过 Internet 发送它们。CloudWatch Logs 支持所有区域的 IPv4 VPC 端点，并且支持除亚太地区 (马来西亚)、亚太地区 (泰国) 和墨西哥 (中部) 以外的所有区域的 IPv6 端点。

Amazon VPC 是一项 Amazon 服务，可用来启动在虚拟网络中定义的 Amazon 资源。借助 VPC，您可以控制您的网络设置，如 IP 地址范围、子网、路由表和网络网关。要将 VPC 连接到 CloudWatch Logs，请为 CloudWatch Logs 定义一个接口 VPC 终端节点。这种类型的端点使您能够将 VPC 连接到 Amazon 服务。该端点提供了到 CloudWatch Logs 的可靠、可扩展的连接，无需互联网网关、网络地址转换 (NAT) 实例或 VPN 连接。有关更多信息，请参阅 Amazon VPC 用户指南中的[什么是 Amazon VPC](#)。

接口 VPC 终端节点由 Amazon PrivateLink 提供支持，后者是一种 Amazon 技术，可将弹性网络接口与私有 IP 地址结合使用来支持 Amazon 服务之间的私有通信。有关更多信息，请参阅[新功能 – Amazon 服务的 Amazon PrivateLink](#)。

以下步骤适用于 Amazon VPC 的用户。有关更多信息，请参阅 Amazon VPC 用户指南中的[入门](#)。

可用性

CloudWatch Logs 当前在所有 Amazon 区域支持 VPC 端点，包括 Amazon GovCloud (US) 区域。

为 CloudWatch Logs 创建 VPC 终端节点

要开始使用 CloudWatch Logs 与 VPC，请为 CloudWatch Logs 创建一个接口 VPC 终端节点。要选择的服务是 `com.amazonaws.Region.logs`。您无需更改 CloudWatch Logs 的任何设置。有关更多信息，请参阅 Amazon VPC 用户指南中的[创建接口端点](#)。

测试 VPC 和 CloudWatch Logs 之间的连接

创建端点后，您可以测试连接。

测试 VPC 和 CloudWatch Logs 端点之间的连接

1. 连接到位于您 VPC 中的 Amazon EC2 实例。有关连接的信息，请参阅 Amazon EC2 文档中的[连接到您的 Linux 实例](#)或[连接到您的 Windows 实例](#)。
2. 在实例中，使用 Amazon CLI 在您的一个现有日志组中创建日志条目。

首先，创建一个包含日志事件的 JSON 文件。必须将时间戳指定为自 1970 年 1 月 1 日 00:00:00 UTC 之后的毫秒数。

```
[
  {
    "timestamp": 1533854071310,
    "message": "VPC Connection Test"
  }
]
```

然后使用 `put-log-events` 命令创建日志条目：

```
aws logs put-log-events --log-group-name LogGroupName --log-stream-  
name LogStreamName --log-events file://JSONFileName
```

如果对该命令的响应包含 `nextSequenceToken`，则该命令已成功执行，并且您的 VPC 终端节点正常运行。

控制对 CloudWatch Logs VPC 终端节点的访问

VPC 终端节点策略是一种 IAM 资源策略，您在创建或修改端点时可将它附加到端点。如果在创建端点时未附加策略，我们将为您附加默认策略以允许对服务进行完全访问。端点策略不会覆盖或替换 IAM 策略或服务特定的策略。这是一个单独的策略，用于控制从端点中对指定服务进行的访问。

端点策略必须采用 JSON 格式编写。

有关更多信息，请参阅《Amazon VPC 用户指南》中的[使用 VPC 端点控制对服务的访问权限](#)。

下面是用于 CloudWatch Logs 的端点策略示例。该策略允许通过 VPC 连接到 CloudWatch Logs 的用户创建日志流并将日志发送到 CloudWatch Logs，并禁止他们执行其他 CloudWatch Logs 操作。

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

修改 CloudWatch Logs 的 VPC 终端节点策略

1. 通过 <https://console.aws.amazon.com/vpc/> 打开 Amazon VPC 控制台。
2. 在导航窗格中，选择 Endpoints (端点)。
3. 如果还没有为 CloudWatch Logs 创建端点，请选择 Create Endpoint (创建端点)。接下来，选择 com.amazonaws.**Region**.logs，然后选择 Create endpoint (创建端点)。
4. 选择 com.amazonaws.**Region**.logs 端点，然后在屏幕下半部分中选择 Policy (策略) 选项卡。
5. 选择 Edit Policy (编辑策略) 并对策略进行更改。

对 VPC 上下文键的支持

CloudWatch Logs 支持可用于限制对特定 VPC 或特定 VPC 终端节点进行访问的 `aws:SourceVpc` 和 `aws:SourceVpce` 上下文键。这些键仅当用户使用 VPC 终端节点时才起作用。有关更多信息，请参阅 IAM 用户指南中的 [可用于部分服务的键](#)。

在 Amazon CloudTrail 中记录 CloudWatch Logs API 和控制台操作

Amazon CloudWatch Logs 与 [Amazon CloudTrail](#) 集成，后者是一项提供用户、角色或 Amazon Web Services 服务所采取操作的记录的服务。CloudTrail 将 CloudWatch Logs 的 API 调用作为事件捕获。捕获的调用包括来自 CloudWatch Logs 控制台的调用和对 CloudWatch Logs API 操作的代码调用。借助 CloudTrail 收集的信息，您可以确定向 CloudWatch Logs 发出的请求、发出请求的 IP 地址、请求的发出时间及其他详细信息。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是用户凭证发出的。
- 请求是否代表 IAM Identity Center 用户发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 Amazon Web Services 服务发出。

当您创建账户并可以自动访问 CloudTrail 事件历史记录时，CloudTrail 在您的 Amazon Web Services 账户中处于活动状态。CloudTrail 事件历史记录提供对 Amazon Web Services 区域中过去 90 天的已记录管理事件的可查看、可搜索、可下载和不可变记录。有关更多信息，请参见《Amazon CloudTrail 用户指南》的 [使用 CloudTrail 事件历史记录](#)。查看事件历史记录不会收取 CloudTrail 费用。

要持续记录您的 Amazon Web Services 账户过去 90 天的事件，请创建跟踪或 [CloudTrail Lake](#) 事件数据存储。

CloudTrail 跟踪

通过跟踪记录，CloudTrail 可将日志文件传送至 Simple Storage Service (Amazon S3) 存储桶。使用 Amazon Web Services 管理控制台创建的所有跟踪均具有多区域属性。您可以通过使用 Amazon CLI 创建单区域或多区域跟踪。建议创建多区域跟踪，因为您可记录您账户中的所有 Amazon Web Services 区域的活动。如果您创建单区域跟踪，则只能查看跟踪的 Amazon Web Services 区域中记录的事件。有关跟踪的更多信息，请参阅《Amazon CloudTrail 用户指南》中的[为您的 Amazon Web Services 账户创建跟踪](#)和[为组织创建跟踪](#)。

通过创建跟踪，您可以从 CloudTrail 免费向您的 Amazon S3 存储桶传送一份正在进行的管理事件的副本，但会收取 Amazon S3 存储费用。有关 CloudTrail 定价的更多信息，请参阅 [Amazon CloudTrail 定价](#)。有关 Amazon S3 定价的信息，请参阅 [Amazon S3 定价](#)。

CloudTrail Lake 事件数据存储

CloudTrail Lake 允许您对事件运行基于 SQL 的查询。CloudTrail Lake 可将基于行的 JSON 格式的现有事件转换为 [Apache ORC](#) 格式。ORC 是一种针对快速检索数据进行优化的列式存储格式。事件将被聚合到事件数据存储中，它是基于您通过应用[高级事件选择器](#)选择的条件的不可变的事件集合。应用于事件数据存储的选择器用于控制哪些事件持续存在并可供您查询。有关 CloudTrail Lake 的更多信息，请参阅《Amazon CloudTrail 用户指南》中的[使用 Amazon CloudTrail Lake](#)。

CloudTrail Lake 事件数据存储和查询会产生费用。创建事件数据存储时，您可以选择要用于事件数据存储的[定价选项](#)。定价选项决定了摄取和存储事件的成本，以及事件数据存储的默认和最长保留期。有关 CloudTrail 定价的更多信息，请参阅 [Amazon CloudTrail 定价](#)。

CloudWatch Logs 支持将以下操作记录为 CloudTrail 日志文件中的事件：

- [AssociateKmsKey](#)
- [CancelExportTask](#)
- [CreateDelivery](#)
- [CreateExportTask](#)
- [CreateLogAnomalyDetector](#)
- [CreateLogGroup](#)
- [CreateLogStream](#)
- [DeleteAccountPolicy](#)
- [DeleteDataProtectionPolicy](#)
- [DeleteDelivery](#)
- [DeleteDeliveryDestination](#)
- [DeleteDeliveryDestinationPolicy](#)
- [DeleteDeliverySource](#)
- [DeleteDestination](#)
- [DeleteIndexPolicy](#)
- [DeleteIntegration](#)
- [DeleteLogAnomalyDetector](#)
- [DeleteLogGroup](#)
- [DeleteLogStream](#)

- [DeleteMetricFilter](#)
- [DeleteQueryDefinition](#)
- [DeleteResourcePolicy](#)
- [DeleteRetentionPolicy](#)
- [DeleteSubscriptionFilter](#)
- [DeleteTransformer](#)
- [DescribeAccountPolicies](#)
- [DescribeConfigurationTemplates](#)
- [DescribeDeliveries](#)
- [DescribeDeliveryDestinations](#)
- [DescribeDeliverySources](#)
- [DescribeDestinations](#)
- [DescribeExportTasks](#)
- [DescribeFieldIndexes](#)
- [DescribeIndexPolicies](#)
- [DescribeLogGroups](#)
- [DescribeLogStreams](#)
- [DescribeMetricFilters](#)
- [DescribeQueries](#)
- [DescribeQueryDefinitions](#)
- [DescribeResourcePolicies](#)
- [DescribeSubscriptionFilters](#)
- [DisassociateKmsKey](#)
- [FilterLogEvents](#)
- [GetDataProtectionPolicy](#)
- [GetDelivery](#)
- [GetDeliveryDestination](#)
- [GetDeliveryDestinationPolicy](#)
- [GetDeliverySource](#)
- [GetIntegration](#)

- [GetLogAnomalyDetector](#)
- [GetLogEvents](#)
- [GetLogGroupFields](#)
- [GetLogRecord](#)
- [GetQueryResults](#)
- [GetTransformer](#)
- [ListAnomalies](#)
- [ListIntegrations](#)
- [ListLogAnomalyDetectors](#)
- [ListLogGroups](#)
- [ListLogGroupsForQuery](#)
- [ListTagsForResource](#)
- [ListTagsLogGroup](#)
- [PutAccountPolicy](#)
- [PutDataProtectionPolicy](#)
- [PutDeliveryDestination](#)
- [PutDeliveryDestinationPolicy](#)
- [PutDeliverySource](#)
- [PutDestination](#)
- [PutDestinationPolicy](#)
- [PutIndexPolicy](#)
- [PutIntegration](#)
- [PutMetricFilter](#)
- [PutQueryDefinition](#)
- [PutResourcePolicy](#)
- [PutRetentionPolicy](#)
- [PutSubscriptionFilter](#)
- [PutTransformer](#)
- [StartLiveTail](#)
- [StartQuery](#)

- [StopQuery](#)
- [TagResource](#)
- [TestMetricFilter](#)
- [TestTransformer](#)
- [UntagResource](#)
- [UpdateAnomaly](#)
- [UpdateDeliveryConfiguration](#)
- [UpdateLogAnomalyDetector](#)

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 Amazon 服务发出。

有关更多信息，请参阅 [CloudTrail userIdentity 元素](#)。

CloudTrail 中的查询生成信息

还支持查询生成器控制台事件的 CloudTrail 日志记录。查询生成器目前支持 CloudWatch Logs Insights 和 CloudWatch Metric Insights。在这些 CloudTrail 事件中，eventSource 为 `monitoring.amazonaws.com`。

以下示例显示了一个 CloudTrail 日志条目，该条目演示了 CloudWatch Logs Insights 中的 `GenerateQuery` 操作。

```
{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
```

```

        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111222333444:role/Administrator",
        "accountId": "123456789012",
        "userName": "SAMPLE_NAME"
    },
    "attributes": {
        "creationDate": "2020-04-08T21:43:24Z",
        "mfaAuthenticated": "false"
    }
}
},
"eventTime": "2020-04-08T23:06:30Z",
"eventSource": "monitoring.amazonaws.com",
"eventName": "GenerateQuery",
"awsRegion": "us-east-1",
"sourceIPAddress": "127.0.0.1",
"userAgent": "exampleUserAgent",
"requestParameters": {
    "query_ask": "****",
    "query_type": "LogsInsights",
    "logs_insights": {
        "fields": "****",
        "log_group_names": ["yourloggroup"]
    },
    "include_description": true
},
"responseElements": null,
"requestID": "2f56318c-cfbd-4b60-9d93-1234567890",
"eventID": "52723fd9-4a54-478c-ac55-1234567890",
"readOnly": true,
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management"
}

```

了解日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日记账条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

以下日志文件条目显示某个用户调用了 CloudWatch Logs CreateExportTask 操作。

```
{
  "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"
}
```

使用 CloudWatch 指标进行监控

您可以使用本节中的表来查看 Amazon CloudWatch Logs 每分钟发送到 Amazon CloudWatch 的指标。

对于跨多个账户和区域的集中监控，您可以使用 CloudWatch Logs 集中化将日志数据和指标集中整合到一个中心位置。有关更多信息，请参阅 [跨账户跨区域日志集中化](#)。

CloudWatch Logs 指标

AWS/Logs 命名空间包括以下指标。

指标	描述
CallCount	在您账户中执行的指定 API 操作的数量。 CallCount 为 CloudWatch Logs 服务使用量指标。有关更多信息，请参阅 CloudWatch Logs 服务使用情况指标。 有效维度：类、资源、服务、类型 有效统计数据：Sum 单位：无
DeliveryErrors	在将数据转发到订阅目标时，CloudWatch Logs 收到其错误的日志事件的数量。如果目标服务返回一个可重试错误，如限制异常或可重试服务异常（如 HTTP 5xx），则 CloudWatch Logs 将继续重试传输长达 24 小时。如果错误是不可重试错误（例如 AccessDeniedException 或 ResourceNotFoundException），则 CloudWatch Logs 不会尝试重试传输。 有效维度：LogGroupName、DestinationType、FilterName、PolicyLevel 有效统计数据：Sum 单位：无
DeliveryThrottling	在将数据转发到订阅目标时，CloudWatch Logs 因为其而被限制的日志事件的数量。

指标	描述
	如果目标服务返回一个可重试错误，如限制异常或可重试服务异常（如 HTTP 5xx），则 CloudWatch Logs 将继续重试传输长达 24 小时。如果错误是不可重试错误（例如 <code>AccessDeniedException</code> 或 <code>ResourceNotFoundException</code>），则 CloudWatch Logs 不会尝试重试传输。 有效维度：LogGroupName、DestinationType、FilterName、PolicyLevel 有效统计数据：Sum 单位：无
EMFDisabledErrors	由于与日志组匹配的 METRIC_EXTRACTION_POLICY 类型的活动账户策略而被忽略的 EMF 格式日志事件的数量。有关指标提取策略的更多信息，请参阅 PutAccountPolicy。 有效维度：LogGroupName 有效统计数据：Sum 单位：无
EMFParsingErrors	处理嵌入式指标格式日志时遇到的解析错误数量。当日志被识别为嵌入式指标格式但不遵循正确的格式时，就会发生此类错误。有关嵌入式指标格式的更多信息，请参阅规范：嵌入式指标格式。 有效维度：LogGroupName 有效统计数据：Sum 单位：无

指标	描述
EMFValidationErrors	处理嵌入式指标格式日志时遇到的验证错误数量。当嵌入式指标格式日志中的指标定义不遵循嵌入式指标格式和 MetricDatum 规范时，就会出现这些错误。有关 CloudWatch 嵌入式指标格式的信息，请参阅规范：嵌入式指标格式。有关数据类型 MetricDatum 的信息，请参阅 Amazon CloudWatch API 参考中的 MetricDatum。  Note 某些验证错误可能导致 EMF 日志中的多个指标未发布。例如，所有使用无效命名空间设置的指标均将被删除。 有效维度：LogGroupName 有效统计数据：Sum 单位：无
ErrorCount	在您账户中执行的导致了错误的 API 操作的数量。 ErrorCount 为 CloudWatch Logs 服务使用量指标。有关更多信息，请参阅 CloudWatch Logs 服务使用情况指标。 有效维度：类、资源、服务、类型 有效统计数据：Sum 单位：无
ForwardedBytes	以压缩字节转发到订阅目标的日志事件的容量。 有效维度：LogGroupName、DestinationType、FilterName 有效统计数据：Sum 单位：字节

指标	描述
ForwardedLogEvents	转发到订阅目标的日志事件的数量。 有效维度：LogGroupName、DestinationType、FilterName、PolicyLevel 有效统计数据：Sum 单位：无
IncomingBytes	以未压缩字节上传到 CloudWatch Logs 的日志事件的容量。当用于 LogGroupName 维度时，这是以未压缩字节上传到日志组的日志事件的容量。 有效维度：LogGroupName 有效统计数据：Sum 单位：字节
IncomingLogEvents	上传到 CloudWatch Logs 的日志事件的数量。当用于 LogGroupName 维度时，这是上传到日志组的日志事件的数量。 有效维度：LogGroupName 有效统计数据：Sum 单位：无
LogEventsWithFindings	与您正使用 CloudWatch Logs 数据保护功能审计的数据字符串相匹配的日志事件数量。有关更多信息，请参阅 通过屏蔽帮助保护敏感的日志数据。 有效维度：无 有效统计数据：Sum 单位：无

指标	描述
ThrottleCount	因使用量配额而被限制在您账户中执行的 API 操作数量。 ThrottleCount 为 CloudWatch Logs 服务使用量指标。有关更多信息，请参阅 CloudWatch Logs 服务使用情况指标。 有效维度：类、资源、服务、类型 有效统计数据：Sum 单位：无

CloudWatch Logs 指标的维度

下表列出了可与大多数 CloudWatch Logs 指标一起使用的维度。

维度	描述
LogGroupName	要显示其指标的 CloudWatch Logs 日志组的名称。
DestinationType	CloudWatch Logs 数据的订阅目标，可以是 Amazon Lambda、Amazon Kinesis Data Streams 或 Amazon Data Firehose。
FilterName	将数据从日志组转发到目标的订阅筛选器的名称。订阅筛选条件名称将由 CloudWatch 自动转换为 ASCII，且任何不受支持的字符将被替换为问号 (?)。

订阅筛选指标维度

下表列出了与账户级订阅筛选条件相关的指标维度。

维度	描述
PolicyLevel	策略适用的级别。目前，此维度的唯一有效值是 AccountPolicy

维度	描述
DestinationType	CloudWatch Logs 数据的订阅目标，可以是 Amazon Lambda、Amazon Kinesis Data Streams 或 Amazon Data Firehose。
FilterName	将数据从日志组转发到目标的订阅筛选器的名称。订阅筛选条件名称将由 CloudWatch 自动转换为 ASCII，且任何不受支持的字符将被替换为问号 (?)。

日志转换器指标和维度

CloudWatch Logs 将以下日志转换器指标发布到 AWS/Logs 命名空间中的 CloudWatch。

指标	描述
TransformationErrors	使用指定转换器转换日志事件时遇到的错误数量。 单位：无 有效统计数据：总计
TransformedBytes	转换后的日志事件的输出量，以未压缩字节为单位。 单位：字节 有效统计数据：总计
TransformedLogEvents	转换后的日志事件数量。 单位：无 有效统计数据：总计

转换器指标会使用以下维度。

维度	描述
LogGroupName	此维度仅用于日志组级转换器。
PolicyLevel	此维度仅用于账户级转换器。目前，此维度的唯一有效值是 AccountPolicy

CloudWatch Logs 服务使用情况指标

CloudWatch Logs 将指标发送到 CloudWatch，用于跟踪 CloudWatch Logs API 操作的使用情况。这些指标与 Amazon 服务限额对应。跟踪这些指标可帮助您主动管理限额。有关更多信息，请参阅 [Service Quotas 集成和使用量指标](#)。

例如，您可以跟踪 ThrottleCount 指标或为该指标设置警报。如果该指标的值上升，则应考虑为受到限制的 API 操作请求增加配额。有关 CloudWatch Logs 服务配额的更多信息，请参阅 [CloudWatch Logs 配额](#)。

CloudWatch Logs 在 AWS/Usage 和 AWS/Logs 命名空间中每分钟发布一次服务配额使用量指标。

下表列出了 CloudWatch Logs 所发布的服务使用量指标。这些指标没有指定的单位。指标中最实用的统计数据是 SUM，它表示以 1 分钟为间隔的总操作数。

这些指标中的每个指标都会发布，其中包含所有 Service、Class、Type 和 Resource 维度的值。它们还与一个名为 Account Metrics 的单一维度一同发布。使用 Account Metrics 维度来查看您账户中所有 API 操作的指标总和。使用其他维度并指定 Resource 维度的 API 操作名称，以查找该特定 API 的指标。

Metrics

指标	描述
CallCount	在您的账户中执行的指定操作的数量。 CallCount 同时发布在 AWS/Usage 和 AWS/Logs 命名空间中。
ErrorCount	在您账户中执行的导致了错误的 API 操作的数量。 ErrorCount 仅发布在 中。AWS/Logs

指标	描述
ThrottleCount	因使用量配额而被限制在您账户中执行的 API 操作数量。 ThrottleCount 仅发布在 中。AWS/Logs

Dimensions

维度	描述
Account metrics	使用该维度可获得所有 CloudWatch Logs API 的指标的总和。 如果要查看一个特定 API 的指标，请使用此表中列出的其他维度，并将该 API 名称指定为 Resource 的值。
Service	包含该资源的 Amazon 服务的名称。对于 CloudWatch Logs 使用情况指标，此维度的值为 Logs。
Class	所跟踪的资源的类。CloudWatch Logs API 使用情况指标使用此维度（值为 None）。
Type	所跟踪的资源类型。目前，当 Service 维度为 Logs 时，Type 的唯一有效值为 API。
Resource	API 操作的名称。有效值包括 操作 中列出的所有 API 操作名称。例如，PutLogEvents

CloudWatch Logs 配额

您可以使用本节中的表来查看 Amazon CloudWatch Logs 中 Amazon 账户的默认服务配额（也称为限制）。大多数服务配额（但不是全部）都列在 Service Quotas 控制台的 Amazon CloudWatch Logs 命名空间下。

Note

如果想要请求增加任何这些配额的配额，请参阅本节中的[过程](#)。

名称	默认值	可调整	描述
活动导出任务	每个受支持的区域：1 个	否	每个账户的活动（正在运行或待处理）导出任务的数量
AssociateKmsKey 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 associate-kms-key 调用的最大次数
批次大小	每个受支持的区域：1MB	否	put-log-events 请求的最大批处理大小（以 MB 为单位）
CancelExportTask 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 cancel-export-task 调用的最大次数
CreateExportTask 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 create-export-task 调用的最大次数

名称	默认值	可调整	描述
CreateLogGroup 每秒事务数节流限制	每个受支持的区域：每秒 10 个	是	每个账户/每个区域每秒 create-log-group 调用的最大次数
CreateLogStream 每秒事务数节流限制	每个受支持的区域：每秒 50 个	是	每个账户/每个区域每秒 create-log-stream 调用的最大次数
数据存档	每个受支持的区域：5 GB	否	免费数据存档的最大大小（以 GB 为单位）
DeleteDestination 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 delete-destination 调用的最大次数
DeleteLogGroup 每秒事务数节流限制	每个受支持的区域：每秒 10 个	是	每个账户/每个区域每秒 delete-log-group 调用的最大次数
DeleteLogStream 每秒事务数节流限制	每个受支持的区域：每秒 15 个	是	每个账户/每个区域每秒 delete-log-stream 调用的最大次数
DeleteMetricFilter 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 delete-metric-filter 调用的最大次数
DeleteRetentionPolicy 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 delete-retention-policy 调用的最大次数
DeleteSubscriptionFilter 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 delete-subscription-filter 调用的最大次数

名称	默认值	可调整	描述
DeleteTransformer 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 delete-transformer 调用的最大次数
DescribeDestinations 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 describe-destinations 调用的最大次数
DescribeExportTasks 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 describe-export-tasks 调用的最大次数
DescribeLogGroups 每秒事务数节流限制	每个受支持的区域：每秒 10 个	是	每个账户/每个区域每秒 describe-log-groups 调用的最大次数
DescribeLogStreams 每秒事务数节流限制	每个受支持的区域：每秒 25 个	是	每个账户/每个区域每秒 describe-log-streams 调用的最大次数
DescribeMetricFilters 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 describe-metric-filters 调用的最大次数
DescribeSubscriptionFilters 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 describe-subscription-filters 调用的最大次数
事件大小	每个支持的区域：1024 KB	否	最大日志事件大小（以 KB 为单位）

名称	默认值	可调整	描述
FilterLogEvents 每秒事务数节流限制	us-east-1 : 每秒 25 次 ap-northeast-3 : 每秒 5 次 ap-southeast-3 : 每秒 5 个 eu-central-1 : 每秒 5 次 每个其他支持的区域 : 每秒 10 次	否	每个账户/每个区域每秒 filter-log-events 调用的最大次数
GetLogEvents 每秒事务数节流限制	us-west-2 : 每秒 10 次 ap-northeast-3 : 每秒 10 次 ap-southeast-3 : 每秒 10 个 eu-central-1 : 每秒 10 次 eu-west-1 : 每秒 10 次 eu-west-3 : 每秒 30 次 每个其他支持的区域 : 每秒 25 次	否	每个账户/每个区域每秒 get-log-events 调用的最大次数

名称	默认值	可调整	描述
GetQueryResults 每秒事务数节流限制	每个受支持的区域：5 个	否	每个账户/每个区域每秒 get-query-results 调用的最大次数
GetTransformer 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 get-transformer 调用的最大次数
每秒事务中的 ListTagsForResource 节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 list-tags-for-resource 调用的最大次数
ListTagsLogGroup 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 list-tags-log-group 调用的最大次数
Live Tail 并发会话限制	每个支持的区域：15 个	是	每个账户的并发活动 Live Tail 会话的最大数量
日志组	每个受支持的区域：100 万个	是	一个账户可以拥有的日志组的最大数量
每个 Live Tail 会话扫描的日志组数	每个受支持的区域：10 个	否	每个 Live Tail 会话可以扫描的日志组的最大数量
每个日志组的指标筛选条件	每个受支持的区域：100 个	否	每个日志组的指标筛选条件的数量
PutDestination 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 put-destination 调用的最大次数
PutDestinationPolicy 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 put-destination-policy 调用的最大次数

名称	默认值	可调整	描述
PutLogEvents 每秒事务数节流限制	每个支持的区域： 每秒 5000 个	是	每个账户/每个区域每秒 put-log-events 调用的最大次数
PutMetricFilter 每秒事务数节流限制	每个受支持的区域： 每秒 5 个	否	每个账户/每个区域每秒 put-metric-filter 调用的最大次数
PutRetentionPolicy 每秒事务数节流限制	每个受支持的区域： 每秒 5 个	否	每个账户/每个区域每秒 put-retention-policy 调用的最大次数
PutSubscriptionFilter 每秒事务数节流限制	每个受支持的区域： 每秒 5 个	否	每个账户/每个区域每秒 put-subscription-filter 调用的最大次数
PutTransformer 每秒事务数节流限制	每个受支持的区域： 每秒 5 个	否	每个账户/每个区域每秒 put-transformer 调用的最大次数
资源策略	每个受支持的区域： 10 个	否	每个账户/每个区域的资源策略最大数量
StartLiveTail 每秒事务数节流限制	每个受支持的区域： 5 个	否	每个账户/每个区域每秒 start-live-tail 调用的最大次数。此限制独立适用于控制台和 API
StartQuery 每秒事务数节流限制	每个受支持的区域： 5 个	否	每个账户/每个区域每秒 start-query 调用的最大次数
每个日志组的订阅筛选条件	每个受支持的区域： 2 个	否	每个日志组的订阅筛选条件的数量

名称	默认值	可调整	描述
TagLogGroup 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 tag-log-group 调用的最大次数
每秒事务中的 TagResource 节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 tag-resource 调用的最大次数
TestMetricFilter 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 test-metric-filter 调用的最大次数
TestTransformer 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 test-transformer 调用的最大次数
UntagLogGroup 每秒事务数节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 untag-log-group 调用的最大次数
每秒事务中的 UntagResource 节流限制	每个受支持的区域：每秒 5 个	否	每个账户/每个区域每秒 untag-resource 调用的最大次数
日志异常检测器数	每个受支持的区域：500 个	<u>是</u>	活动日志异常检测器的最大数量

管理您的 CloudWatch Logs 服务配额

CloudWatch Logs 已与 Service Quotas 集成，后者是一项 Amazon 服务，可让您从中心位置查看和管理您的配额。有关更多信息，请参阅 Service Quotas 用户指南中的[什么是 Service Quotas ?](#)。

通过 Service Quotas 可轻松查找 CloudWatch Logs 服务配额的值。

Amazon Web Services 管理控制台

使用控制台查看 CloudWatch Logs 服务配额

1. 访问 <https://console.aws.amazon.com/servicequotas/>，打开 Service Quotas 控制台。
2. 在导航窗格中，选择 Amazon services (服务)。
3. 从 Amazon services (服务) 列表中，搜索并选择 Amazon CloudWatch Logs。

在 Service quotas 列表中，您可以查看服务配额名称、应用的值（如果该值可用）、Amazon 默认配额以及配额值是否可调整。

4. 要查看有关服务配额的其他信息（如描述），请选择配额名称。
5. （可选）要请求增加配额，请选择要增加的配额，选择 Request quota increase（请求增加配额），输入或选择所需信息，然后选择 Request（请求）。

要使用控制台进一步处理服务配额，请参阅 [Service Quotas 用户指南](#)。要请求提高配额，请参阅 Service Quotas 用户指南中的 [请求增加配额](#)。

Amazon CLI

使用 Amazon CLI 查看 CloudWatch Logs 服务配额

运行以下命令，查看默认 CloudWatch Logs 配额。

```
aws service-quotas list-aws-default-service-quotas \
  --query 'Quotas[*]'.
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
  --service-code logs \
  --output table
```

要使用 Amazon CLI 进一步处理服务配额，请参阅 [Service Quotas Amazon CLI 命令参考](#)。要请求提高配额，请参阅 [Amazon CLI 命令参考](#) 中的 [request-service-quota-increase](#) 命令。

文档历史记录

下表列出了从 2018 年 6 月开始每个版本 CloudWatch Logs 用户指南的重要更改。要获得本文档的更新通知，您可以订阅 RSS 源。

变更	说明	日期
CloudWatch Logs 增加了对跨账户跨区域日志集中化的支持	CloudWatch Logs 增加了对跨账户跨区域日志集中化的支持，使得 CloudWatch Logs 能够将 Amazon Organizations 成员账户中的日志数据收集到中央账户进行分析。有关更多信息，请参阅 跨账户跨区域日志集中化 。	2025 年 9 月 17 日
CloudWatch Logs 异常检测更新了示例策略	CloudWatch Logs 更新了 KMS 示例策略，通过添加 <code>aws:SourceAccount</code> 和 <code>aws:SourceArn</code> 的条件来缩小其范围并提高安全性。有关更多信息，请参阅 使用 Amazon KMS 对异常检测器及其结果进行加密	2025 年 7 月 14 日
CloudWatch Logs Insights 增加了对 Amazon VPC Route Server 日志的支持	CloudWatch Logs Insights 增加了对 Amazon VPC Route Server 日志的支持。有关更多信息，请参阅 启用从 Amazon 服务进行日志记录 。Amazon VPC Route Server 的新日志源 <code>EVENT_LOGS</code> 记录在 PutDeliverySource 中	2025 年 6 月 12 日
CloudWatch Logs Insights 增加了对 Amazon PCS 日志的支持	CloudWatch Logs Insights 增加了对 Amazon PCS 日志的支持。有关更多信息，请参阅 启	2025 年 6 月 12 日

[用从 Amazon 服务进行日志记录](#)。Amazon PCS 的新日志源 PCS_SCHEDULER_LOGS 和 PCS_JOBCOMP_LOGS 记录在 [PutDeliverySource](#) 中

[CloudWatch Logs Insights 增加了对 Amazon Entity Resolution 数据匹配服务 日志的支持](#)

CloudWatch Logs Insights 增加了对 Amazon Entity Resolution 数据匹配服务 日志的支持。有关更多信息，请参阅[启用从 Amazon 服务进行日志记录](#)。Amazon Entity Resolution 数据匹配服务 的新日志源 WORKFLOW_LOGS 记录在 [PutDeliverySource](#) 中

2025 年 5 月 22 日

[CloudWatch Logs 托管策略更新，支持自然语言摘要](#)

cloudwatch:GenerateQueryResultsSummary 的权限已添加到 CloudWatchLogsFullAccess 和 CloudWatchLogsReadOnlyAccess 中，允许生成查询结果的自然语言摘要。要查看策略的内容，请参阅《Amazon 托管策略参考指南》中的 [CloudWatchLogsFullAccess](#) 和 [CloudWatchLogsReadOnlyAccess](#)。

2025 年 5 月 20 日

[CloudWatch Logs Insights 增加了对从 CloudWatch Logs Insights 查询结果生成自然语言摘要的支持](#)

CloudWatch Logs Insights 中增加了对自然语言摘要的支持。此功能会生成用户可读的查询结果摘要，目前在美国东部（弗吉尼亚州北部）可用。有关更多信息，请参阅[根据 CloudWatch Logs Insights 查询结果生成自然语言摘要](#)。

2025 年 5 月 20 日

CloudWatchOpenSearchDashboardsFullAccess 是新的 IAM 策略	CloudWatch Logs 新增了两个 IAM 策略：CloudWatchOpenSearchDashboardsFullAccess 和 CloudWatchOpenSearchDashboardsAccess。CloudWatchOpenSearchDashboardsFullAccess 授予创建和管理与 OpenSearch Service 的集成的权限。CloudWatchOpenSearchDashboardsAccess 策略授予查看在这些集成中创建的公开发布的日志控制板的权限。有关更多信息，请参阅 Vended log dashboards powered by Amazon OpenSearch Service 。	2024 年 12 月 1 日
CloudWatchLogsFullAccess 策略已更新	CloudWatch Logs 已将 Amazon OpenSearch Service 和 IAM 的权限添加到 CloudWatchLogsFullAccess 策略中，以便将 CloudWatch Logs 与 OpenSearch Service 集成，从而实现某些功能。	2024 年 12 月 1 日
CloudWatch Logs Insights 为查询语法添加了新的结构类型	CloudWatch Logs Insights 添加了 unnest 命令和两个 JSON 函数，使您能够将 JSON 字符串作为映射和列表进行操作。有关更多信息，请参阅 结构类型 。	2024 年 11 月 21 日

[CloudWatch Logs 支持在日志摄取期间进行日志转换](#)

您可以创建日志转换器来在摄取时修改日志事件，从而帮助您将不同格式和来自不同来源的日志标准化为一致且上下文丰富的格式。有关更多信息，请参阅[在摄取期间转换日志](#)。

2024 年 11 月 20 日

[CloudWatch Logs Insights 添加了字段索引](#)

CloudWatch Logs Insights 增加了对日志字段索引的支持。然后，当您在 CloudWatch Logs Insights 查询中使用字段索引时，查询会尝试跳过处理已知不包含索引字段的日志事件。有关更多信息，请参阅[创建字段索引以提高查询性能并减少扫描量](#)。

2024 年 11 月 20 日

[CloudWatch Logs Insights 对自然语言查询生成的支持已正式发布](#)

CloudWatch Logs Insights 支持自然语言生成和更新查询。有关更多信息，请参阅[使用自然语言生成和更新 CloudWatch Logs Insights 查询](#)。

2024 年 6 月 20 日

[CloudWatchLogsReadOnlyAccess 策略已更新](#)

CloudWatch Logs 为 CloudWatchLogsReadOnlyAccess 添加了 `cloudwatch:GenerateQuery` 权限，以便具有此策略的用户可以根据自然语言提示生成 [CloudWatch Logs Insights](#) 查询字符串。

2023 年 11 月 26 日

[CloudWatchLogsFullAccess 策略已更新](#)

CloudWatch Logs 为 CloudWatchLogsFullAccess 添加了 `cloudwatch:GenerateQuery` 权限，以便具有此策略的用户可以根据自然语言提示生成 [CloudWatch Logs Insights](#) 查询字符串。

2023 年 11 月 26 日

[CloudWatch Logs 添加了日志模式分析](#)

现在，每次执行 CloudWatch Logs Insights 查询时，Cloud Watch Logs 都会扫描日志事件中的模式。有关更多信息，请参阅[模式分析](#)。

2023 年 11 月 26 日

[CloudWatch Logs 添加了日志异常检测](#)

您可以为日志组创建日志异常检测器。异常检测器会扫描提取到日志组中的日志事件，并发现日志数据中的异常。有关更多信息，请参阅[日志异常检测](#)。

2023 年 11 月 26 日

[CloudWatch Logs 添加了比较功能](#)

现在，您可以使用 CloudWatch Logs Insights 比较一段时间内日志事件的变化。有关更多信息，请参阅[与之前的时间范围进行比较 \(差异\)](#)。

2023 年 11 月 26 日

[CloudWatch Logs 添加了一个新的日志类](#)

CloudWatch Logs 支持两类日志组，以便您可以为不经常访问的日志提供经济高效的选项，同时还可以为需要实时监控或其他功能的日志提供功能齐全的选项。有关更多信息，请参阅[日志类](#)。

2023 年 11 月 26 日

[CloudWatch Metrics Insights 支持自然语言查询生成](#)

CloudWatch Logs Insights 支持自然语言生成和更新查询。有关更多信息，请参阅[使用自然语言生成和更新 CloudWatch Logs Insights 查询](#)。

2023 年 11 月 26 日

[CloudWatch Logs 为 Live Tail 增加了正则表达式筛选条件模式语法支持](#)

现在，您可以在 Live Tail 筛选条件模式中，使用灵活的正则表达式进一步自定义搜索和匹配操作，以满足您的需求。有关更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[筛选条件模式语法](#)。

2023 年 11 月 13 日

[CloudWatch Logs 为指标筛选条件、订阅筛选条件和筛选日志事件增加正则表达式筛选条件模式语法支持](#)

现在，您可以在筛选条件模式中，使用灵活的正则表达式进一步自定义搜索和匹配操作，以满足您的需求。有关更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[筛选条件模式语法](#)。

2023 年 9 月 5 日

[CloudWatch Logs Insights 添加了模式命令](#)

现在，您可以在 CloudWatch Logs Insights 查询中使用模式，将您的日志数据自动划分为不同模式。模式是在日志字段中重复出现的共有的文本结构。有关更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[pattern](#)。

2023 年 7 月 17 日

[CloudWatch Logs Insights 添加了 dedup 命令](#)

现在，您可以在 CloudWatch Logs Insights 查询中使用 dedup，根据您指定的字段中的特定值删除重复结果。有关更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[重复数据删除](#)。

2023 年 6 月 20 日

[账户级别数据保护策略](#)

现在，您可以在账户级别设置数据保护策略。这些账户级别策略可以审计和屏蔽账户中所有日志组的日志事件中的敏感信息。有关更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的[通过屏蔽帮助保护敏感日志数据](#)。

2023 年 6 月 8 日

[添加了 Live Tail 功能](#)

CloudWatch Logs 添加了 Live Tail 功能，因此您可以在提取日志时对其进行扫描，以帮助进行故障排除。您可以选择根据指定术语筛选显示的日志事件流，也可以突出显示具有指定术语的日志事件。有关更多信息，请参阅[使用 live tail 近乎实时地查看日志](#)。

2023 年 6 月 6 日

[CloudWatchLogsRead OnlyAccess 策略已更新](#)

CloudWatch Logs 已向 CloudWatchLogsRead OnlyAccess 添加了权限。添加了 logs:StartLiveTail 和 logs:StopLiveTail 权限，以便使用此策略的用户可以使用控制台启动和停止 CloudWatch Logs live tail 会话。有关更多信息，请参阅 [使用 Live Tail 近乎实时地查看日志](#)。

2023 年 6 月 6 日

[CloudWatch Logs Insights 已发布](#)

您可以使用 CloudWatch Logs Insights 以交互方式搜索和分析您的日志数据。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南中的 [使用 CloudWatch Logs Insights 分析日志数据](#)。

2018 年 11 月 27 日

[对 Amazon VPC 端点的支持](#)

您现在可以在 VPC 和 CloudWatch Logs 之间建立私有连接。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南中的 [将 CloudWatch Logs 与接口 VPC 终端节点结合使用](#)。

2018 年 6 月 28 日

下表介绍了 Amazon CloudWatch Logs 用户指南的重要更改。

更改	描述	发布日期
接口 VPC 端点	在某些区域，您可以使用接口 VPC 终端节点，以防止 Amazon VPC 和 CloudWatch Logs 之间的流量离开 Amazon 网络。有关更多信息，请参阅	2018 年 3 月 7 日

更改	描述	发布日期
	将 CloudWatch Logs 与接口 VPC 终端节点结合使用。	
Route 53 DNS 查询日志	您可以使用 CloudWatch Logs 存储有关 Route 53 收到的 DNS 查询的日志。有关更多信息，请参阅 Amazon Route 53 开发人员指南中的 什么是 Amazon CloudWatch Logs ? 或 记录 DNS 查询 。	2017 年 9 月 7 日
标记日志组	您可以使用标签对日志组进行分类。有关更多信息，请参阅 在 Amazon CloudWatch Logs 中标记日志组 。	2016 年 12 月 13 日
控制台改进	您可以从指标图导航到关联的日志组。有关更多信息，请参阅 从指标定向至日志 。	2016 年 11 月 7 日
控制台可用性改进	改善了体验以更轻松地搜索、筛选和排查问题。例如，您现在可以筛选出某个日期和时间范围内的日志数据。有关更多信息，请参阅 查看发送到 CloudWatch Logs 的日志数据 。	2016 年 8 月 29 日
增加了 Amazon CloudTrail 对 Amazon CloudWatch Logs 和新的 CloudWatch Logs 指标的支持	增加了 Amazon CloudTrail 对 Amazon CloudWatch Logs 的支持。有关更多信息，请参阅 在 Amazon CloudTrail 中记录 CloudWatch Logs API 和控制台操作 。	2016 年 3 月 10 日
增加了对 Amazon CloudWatch Logs 导出到 Amazon S3 的支持	增加了对将 CloudWatch Logs 数据导出到 Amazon S3 的支持。有关更多信息，请参阅 将日志数据导出到 Amazon S3 。	2015 年 12 月 7 日

更改	描述	发布日期
增加了对 Amazon CloudWatch Logs 中 Amazon CloudTrail 记录事件的支持	您可以在 CloudWatch 中创建告警并接收 CloudTrail 捕获的特定 API 活动的通知，然后使用通知执行问题排查。	2014 年 11 月 10 日
增加了对 Amazon CloudWatch Logs 的支持	您可以使用 Amazon CloudWatch Logs 来监控、存储和访问您的系统、应用程序以及来自 Amazon Elastic Compute Cloud (Amazon EC2) 实例或其他来源的自定义日志文件。然后可以使用 Amazon CloudWatch 控制台、Amazon CLI 中的 CloudWatch Logs 命令或使用 CloudWatch Logs 开发工具包，从 CloudWatch Logs 检索关联的日志数据。有关更多信息，请参阅 什么是 Amazon CloudWatch Logs ? 。	2014 年 7 月 10 日

Amazon 术语表

有关最新的 Amazon 术语，请参阅 Amazon Web Services 词汇表 参考中的 [Amazon 词汇表](#)。