aws

用户指南

Amazon CloudWatch 日志

Amazon CloudWatch 日志: 用户指南

Table of Contents

什么是 Amazon CloudWatch 日志？ 1
功能 1
相关 AWS 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
开始使用 7
先决条件 7
注册获取 AWS 账户 7
创建具有管理访问权限的用户 8
设置命令行界面 9
使用统一 CloudWatch 代理 9
使用以前的 CloudWatch 代理 9
CloudWatch 记录代理先决条件 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：在 Windows Server 2016 实例中使用 CloudWatch 日志 20
快速入门：在 Windows Server 2012 和 Windows Server 2008 实例上使用 CloudWatch 日志 31
快速入门：使用安装代理 AWS OpsWorks 40
报告 CloudWatch Logs 代理状态 45
启动 CloudWatch 日志代理 46
停止 CloudWatch Logs 代理 46
快速入门 AWS CloudFormation 47
使用 AWS 软件开发工具包 49
使用 “日志见解” 分析 CloudWatch 日志数据 51
日志类中支持的命令 52
入门：查询教程 52
教程：运行和修改示例查询 53
教程：使用聚合函数运行查询 55
教程：运行生成按日志字段分组的可视化的查询 56
教程：运行生成时间序列可视化的查询 57

支持的日志和发现的字段 57
JSON 日志中的字段 59
查询语法 61
display 63
fields 64
筛选条件 64
模式 67
diff 68
parse 68
sort 70
stats 71
限制 76
dedup 77
unmask 77
布尔值、比较、数值、日期时间和其他函数 77
包含特殊字符的字段 86
在查询中使用别名和注释 86
模式分析 87
模式分析入门 88
有关模式命令的详细信息 90
与之前的时间范围进行比较（差异） 90
示例查询 92
常规查询 93
Lambda 日志的查询 93
Amazon VPC 流日志的查询 94
Route 53 日志的查询 95
查询日 CloudTrail 志 95
查询 Amazon API Gateway 96
NAT 网关的查询 97
查询 Apache 服务器日志 98
针对亚马逊的查询 EventBridge 99
解析命令的示例 99
在图形中可视化日志数据 99
保存并重新运行查询 100
将查询添加到控制面板或导出查询结果 102
查看正在运行的查询或查询历史记录 102

使用加密查询结果 AWS Key Management Service 103
限制 103
步骤 1：创建一个 AWS KMS key 104
步骤 2：设置 KMS 密钥的权限 104
步骤 3：将 KMS 密钥与查询结果关联 106
步骤 4：取消密钥与账户中查询结果的关联 106
使用自然语言生成和更新 L CloudWatch ogs Insights 查询 106
示例查询 107
选择不使用您的数据来改善服务 109
日志异常检测 110
异常和模式的严重性和优先级 110
异常可见时间 111
抑制异常 111
常见问题 111
在日志组上启用异常检测 112
查看已发现的异常 113
在日志异常检测器上创建警报 115
日志异常检测器发布的指标 117
使用对异常检测器及其结果进行加密 AWS KMS 118
限制 118
使用日志组和日志流 122
创建日志组 122
将日志发送到日志组 122
查看日志数据 123
使用 Live Tail 近乎实时地查看日志 123
开始 Live Tail 会话 124
使用筛选条件模式搜索日志数据 126
使用控制台搜索日志条目 126
使用搜索日志条目 AWS CLI 127
从指标定向至日志 127
故障排除 128
更改日志数据保留期 128
标记日志组 129
有关标签的基本知识 129
使用标签跟踪成本 130
标签限制 130

使用标记日志组 AWS CLI 131
使用日志 API 标记 CloudWatch 日志组 131
使用加密日志数据 AWS KMS 132
限制 133
步骤 1：创建 AWS KMS 密钥 104
步骤 2：设置 KMS 密钥的权限 104
步骤 3：将日志组与 KMS 密钥关联 121
步骤 4：取消日志组与密钥的关联 121
KMS 密钥和加密上下文 137
通过屏蔽帮助保护敏感的日志数据 140
了解数据保护策略 142
创建或使用数据保护策略所需的 IAM 权限 145
创建账户范围的数据保护策略 150
为单个日志组创建数据保护策略 153
查看未屏蔽的数据 155
审计结果报告 156
您可以保护的数据类型 157
指标筛选条件 197
概念 198
指标筛选条件的筛选条件模式语法 198
为指标筛选条件配置指标值 200
使用日志事件中的指标发布维度 200
使用录入事件中的值来增加指标的值 203
创建指标筛选条件 204
创建日志组的指标筛选条件 204
示例：对日志事件进行计数 205
示例：对字词的出现次数进行计数 207
示例：对 HTTP 404 代码进行计数 208
示例：对 HTTP 4xx 代码进行计数 211
示例：从 Apache 日志中提取字段并分配维度 212
列出指标筛选条件 214
删除指标筛选条件 215
订阅筛选器 216
概念 217
记录组级订阅过滤器 218
示例 1：Kinesis Data Streams 订阅筛选条件 218

示例 2：带有以下内容的订阅过滤器 AWS Lambda 224
示例 3：使用 Amazon Data Firehose 的订阅筛选条件 227
账户级订阅过滤器 234
示例 1：Kinesis Data Streams 订阅筛选条件 235
示例 2：带有以下内容的订阅过滤器 AWS Lambda 240
示例 3：使用 Amazon Data Firehose 的订阅筛选条件 244
跨账户跨区域订阅 251
使用 Kinesis Data Streams 进行跨账户跨区域日志数据共享 252
使用 Firehose 进行跨账户跨区域日志数据共享 270
使用 Kinesis Data Streams 的跨账户跨区域账户级别订阅 283
使用 Firehose 进行跨账户跨区域账户级别订阅 299
混淆代理问题防范 310
日志递归防护 311
筛选条件模式语法 313
支持的正则表达式 313
使用正则表达式匹配字词 316
匹配非结构化日志事件中的字词 316
匹配 JSON 日志事件中的字词 320
匹配以空格分隔的日志事件中的字词 328
启用来自 AWS 服务的日志记录 332
需要额外权限 [V1] 的日志记录 336
发送到日志的 CloudWatch 日志 337
发送到 Amazon S3 的日志 339
已发送到 Firehose 的日志 342
需要额外权限 [V2] 的日志记录 344
发送到日志的 CloudWatch 日志 345
发送到 Amazon S3 的日志 347
已发送到 Firehose 的日志 351
特定于服务的权限 354
控制台专属权限 354
防止跨服务混淆座席 355
策略更新 356
将日志数据导出到 Amazon S3 358
概念 359
使用控制台将日志数据导出到 Amazon S3 360
同账号导出 360

跨账户导出 366
使用将日志数据导出到 Amazon S3 AWS CLI 374
同账号导出 375
跨账户导出 381
描述导出任务 389
取消导出任务 391
将数据流式传输到 OpenSearch 服务 392
先决条件 392
为日志组订阅 OpenSearch 服务 392
代码示例 394
操作 395
AssociateKmsKey 395
CancelExportTask 397
CreateExportTask 398
CreateLogGroup 400
CreateLogStream 403
DeleteLogGroup 404
DeleteSubscriptionFilter 407
DescribeExportTasks 412
DescribeLogGroups 413
DescribeSubscriptionFilters 417
GetQueryResults 424
PutSubscriptionFilter 425
StartLiveTail 431
StartQuery 443
场景 446
运行大型查询 446
跨服务示例 462
使用计划的事件调用 Lambda 函数 462
安全性 464
数据保护 464
静态加密 465
传输中加密 465
Identity and Access Management 466
身份验证 466
访问控制 466

有关管理访问的概述 467
使用基于身份的策略（IAM 策略） 471
CloudWatch 日志权限参考 482
使用服务相关角色 487
合规性验证 489
故障恢复能力 490
基础设施安全性 490
接口 VPC 端点 490
可用性 491
为 CloudWatch 日志创建 VPC 终端节点 491
测试您的 VPC 和 CloudWatch 日志之间的连接 491
控制对您的 CloudWatch 日志 VPC 终端节点的访问 492
对 VPC 上下文键的支持 493
使用记录 API 和控制台操作 AWS CloudTrail 494
CloudWatch 将信息记录到 CloudTrail 494
查询生成信息 CloudTrail 496
了解日志文件条目 497
代理参考 499
代理配置文件 499
使用带有 HTTP 代理的 CloudWatch 日志代理 505
划分 CloudWatch 日志代理配置文件 506
CloudWatch 日志代理常见问题解答 506
使用 CloudWatch 指标监控使用情况 510
CloudWatch 记录指标 510
CloudWatch 日志指标的维度 513
CloudWatch 记录服务使用情况指标 514
服务限额 517
管理您的 CloudWatch 日志服务配额 521
文档历史记录 523
AWS 词汇表 529
..... dxxx

什么是 Amazon CloudWatch 日志？

您可以使用亚马逊 CloudWatch 日志来监控、存储和访问来自亚马逊弹性计算云 (Amazon EC2) 实例 AWS CloudTrail、Route 53 和其他来源的日志文件。

CloudWatch 日志使您能够将您使用的所有系统、应用程序和 AWS 服务的日志集中到一个高度可扩展的单一服务中。然后，您可以轻松查看它们，在它们中搜索特定的错误代码或模式，根据特定字段对其进行筛选，或者将其安全地存档以备将来分析。 CloudWatch 日志使您可以将所有日志（无论其来源如何）视为按时间排序的单一且一致的事件流。

CloudWatch 日志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，以及使用过滤器或嵌入式日志格式从日志中生成指标。

CloudWatch 日志支持两个日志类别。日志标准CloudWatch 日志类中的日志组支持所有 CloudWatch 日志功能。Log CloudWatch s Infrequent Access 日志类中的日志组产生的摄取费用较低，并且支持标准类功能的子集。有关更多信息，请参阅 日志类。

功能

- 两种日志类别以提高灵活性 — Log CloudWatch s 提供了两个日志类别，因此您可以为不经常访问的日志提供经济实惠的选择。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 Lo CloudWatch gs 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、更有效地应对运营问题。 CloudWatch Logs Insights 包括一种专门构建的查询语言，其中包含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。包括针对几种类型的 AWS 服务日志的示例查询。要开始使用，请参阅 使用“日志见解”分析 CloudWatch 日志数据。
- 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 Live Tail 近乎实时地查看日志。
- 监控来自 Amazon EC2 实例的CloudWatch 日志 — 您可以使用日志通过日志数据监控应用程序和系统。例如， CloudWatch 日志可以跟踪应用程序日志中发生的错误数量，并在错误率超过您指定的阈值时向您发送通知。 CloudWatch 日志使用您的日志数据进行监控；因此，无需更改代码。例如，您可以监视应用程序日志中的特定文字术语（例如 NullReferenceException“”），或者计算文字术语在日志数据中特定位置出现的次数（例如 Apache 访问日志中的“404”状态代码）。找到您要搜索的

术语后， CloudWatch Logs 会将数据报告给您指定的 CloudWatch 指标。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 CloudWatch 日志入门。

- 监控 AWS CloudTrail 记录的事件 — 您可以在中 CloudWatch 创建警报，接收捕获的特定 API 活动的通知， CloudTrail 并使用该通知进行故障排除。要开始使用，请参阅AWS CloudTrail 用户指南中的向CloudWatch 日志发送 CloudTrail 事件。
- 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。
- 存档日志数据-您可以使用 CloudWatch 日志将日志数据存储在高度耐用的存储中。通过 CloudWatch 日志代理，可以轻松地将轮换和非轮换的日志数据从主机快速发送到日志服务。然后，您可以按需访问原始日志数据。
- 记录 Route 53 的 DNS 查询 — 您可以使用 CloudWatch 日志记录有关 Route 53 收到的 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。

相关 AWS 服务

以下服务与 CloudWatch 日志配合使用：

- AWS CloudTrail是一项 Web 服务，可让您监控对账户的 CloudWatch Logs API 进行的调用，包括 AWS Management Console、 AWS Command Line Interface (AWS CLI) 和其他服务发出的调用。开启 CloudTrail 日志记录后，会在您的账户中 CloudTrail捕获 API 调用，并将日志文件传输到您指定的 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。有关的更多信息 AWS CloudTrail，请参阅什么是 AWS CloudTrail？ 在《AWS CloudTrail 用户指南》中。有关 CloudWatch 写入 CloudTrail 日志文件的数据类型的示例，请参见记录 CloudWatch 日志 API 和控制台操作在 AWS CloudTrail。
- AWS Identity and Access Management (IAM) 是一项 Web 服务，可帮助您安全地控制用户对 AWS 资源的访问权限。使用 IAM 可以控制可使用您的 AWS 资源（身份验证）的人员、他们可以使用的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？。

- AWS 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，请参阅什么是 AWS Lambda？ 在《AWS Lambda 开发人员指南》中。

定价

注册后 AWS，您可以使用免费套餐AWS 免费 CloudWatch 开始使用 Logs。

标准费率适用于其他服务使用日志存储的日 CloudWatch 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 日志的成本和使用量的更多信息 CloudWatch，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 日志的概念

下面描述了对您理解和使用 CloudWatch 日志至关重要的术语和概念。

日志类

CloudWatch 日志提供两类日志组。标准日志类是一个功能齐全的选项，适用于需要实时监控的日志或您经常访问的日志。对于访问频率较低的日志，低频访问日志类是一种成本较低的选项。它支持标准日志类功能的子集。

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。Log CloudWatch s 理解的日志事件记录包含两个属性：事件发生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，可以使用 aws lo gs 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器从摄取的事件中提取指标观察值，并将其转换为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

保留设置可用于指定日志事件在CloudWatch 日志中保留多长时间。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AWS后，您可以通过AWS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 日志提供两类日志组：

- Lo CloudWatch gs St andard 日志类是一个功能齐全的选项，适用于需要实时监控的日志或您经常访问的日志。
- Lo CloudWatch gs Infrequent Acces s 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。该日志类提供了一部分 CloudWatch 日志功能，包括托管提取、存储、跨账户日志分析和加密，且每 GB 的摄取价格较低。Infrequent Access 日志类非常适合对不经常访问的日志进行临时查询和 after-the-fact 取证分析。

> **Note**
>
> 在收费方面，标准和不频繁访问日志类别的区别仅在于摄取成本。每个 CloudWatch 日志类别的存储费用和 Logs Insights 费用相同。

有关 CloudWatch 日志定价的更多信息，请参阅 Amazon CloudWatch 定价。

> **Important**
>
> 创建日志组后，无法更改其日志类别。

支持的特征

下表列出了每个日志类的功能。

	Standard	不频繁访问	
完全托管的日志摄取和存储	✓	✓	
跨账户功能	✓	✓	
使用加密 AWS KMS	✓	✓	

	Standard	不频繁访问	
CloudWatch 日志见解查询命令	✓	✓（大多数命令 — 请参阅日志类中支持的命令。）	
CloudWatch 日志见解已发现字段	✓		
自然语言查询帮助	✓		
CloudWatch 日志异常检测	✓		
与之前的时间范围比较	✓		
订阅过滤器	✓		
导出到 Amazon S3	✓		
GetLogEvents和 FilterLogEventsAPI 操作	✓	不支持。使用 Lo CloudWatch gs Insights 查看存储在不频繁访问日志类的日志组中的日志事件。	
指标筛选器	✓		
容器洞察日志提取	✓		
Lambda Insights 日志提取	✓		
使用屏蔽保护敏感数据	✓		
嵌入式指标格式	✓		

CloudWatch 日志入门

要将来自您的 Amazon EC2 实例和本地服务器的日志收集到 CloudWatch 日志中，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。

如果您使用统一 CloudWatch 代理来收集 CloudWatch 指标，则它允许收集其他系统指标，以实现访客内部的可见性。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。

旧的 L CloudWatch ogs 代理仅支持从运行 Linux 的服务器收集日志，现已弃用且不再受支持。有关从较旧的 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。

内容

- 先决条件
- 使用统一 CloudWatch 代理开始使用 CloudWatch Logs
- 使用以前的 CloudWatch 代理开始使用 CloudWatch Logs
- 快速入门： AWS CloudFormation 用于开始使用 CloudWatch 日志

先决条件

要使用 Amazon CloudWatch 日志，您需要一个 AWS 账户。您的 AWS 账户允许您使用服务（例如 Amazon EC2）生成日志，您可以在 CloudWatch 控制台（基于 Web 的界面）中查看这些日志。此外，您还可以安装和配置 AWS Command Line Interface (AWS CLI)。

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，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。

AWS 注册过程完成后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。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

设置命令行界面

您可以使用 AWS CLI 来执行 CloudWatch 日志操作。

有关如何安装和配置的信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》中的使用 AWS 命令行界面进行设置。

使用统一 CloudWatch 代理开始使用 CloudWatch Logs

有关使用统一 CloudWatch 代理开始使用 CloudWatch 日志的更多信息，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。如果您不使用代理来收集 CloudWatch 指标，则可以忽略任何引用指标的部分。

如果您当前使用的是较旧的 CloudWatch Logs 代理，并且想要迁移到使用新的统一代理，我们建议您使用新代理程序包中包含的向导。该向导可以读取您当前的 CloudWatch Logs Agent 配置文件，并将 CloudWatch代理设置为收集相同的日志。有关向导的更多信息，请参阅 Amazon CloudWatch 用户指南中的使用向导创建 CloudWatch 代理配置文件。

使用以前的 CloudWatch 代理开始使用 CloudWatch Logs

> **Important**
>
> CloudWatch 包括一个可以从 EC2 实例和本地服务器收集日志和指标的统一 CloudWatch 代理。仅限日志的较旧代理现已弃用，且不再受支持。

有关从较旧的仅限日志的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。
本节的其余部分将介绍如何为仍在使用旧版 CloudWatch 的 Logs 代理的客户使用该代理。

使用 CloudWatch 日志代理，您可以发布来自运行 Linux 或 Windows Server 的 Amazon EC2 实例的日志数据，以及从中记录的事件 AWS CloudTrail。我们建议改用 CloudWatch 统一代理来发布您的日志数据。有关新代理的更多信息，请参阅 Amazon CloudWatch 用户指南中的使用 CloudWatch 代理从 Amazon EC2 实例和本地服务器收集指标和日志。

内容

- CloudWatch 记录代理先决条件
- 快速入门：在正在运行的 EC2 Linux 实例上安装和配置 CloudWatch 日志代理
- 快速入门：启动时在 EC2 Linux 实例上安装和配置 CloudWatch 日志代理
- 快速入门：允许运行 Windows Server 2016 的 Amazon EC2 实例使用 CloudWatch 日志代理向 CloudWatch 日志发送日志
- 快速入门：允许运行 Windows Server 2012 和 Windows Server 2008 的 Amazon EC2 实例向日志发送 CloudWatch 日志
- 快速入门：使用 AWS OpsWorks 和 Chef 安装 CloudWatch Logs 代理
- 报告 CloudWatch Logs 代理状态
- 启动 CloudWatch 日志代理
- 停止 CloudWatch Logs 代理

CloudWatch 记录代理先决条件

CloudWatch 日志代理需要 Python 版本 2.7、3.0 或 3.3 以及以下任何版本的 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：在正在运行的 EC2 Linux 实例上安装和配置 CloudWatch 日志代理

Important

已弃用较旧的日志代理。 CloudWatch 包括一个可以从 EC2 实例和本地服务器收集日志和指标的统一代理。有关更多信息，请参阅 CloudWatch 日志入门。
有关从较旧的 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。
较早版本的日志代理仅支持 Python 2.6 至 3.5 版本。此外，较旧的 CloudWatch 日志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，则必须使用较新的统一代理而不是较旧 CloudWatch 的 Logs 代理。
本节的其余部分将介绍如何为仍在使用旧版 CloudWatch 的 Logs 代理的客户使用该代理。

Tip

CloudWatch 包括一个新的统一代理，可以从 EC2 实例和本地服务器收集日志和指标。如果您尚未使用较旧的 L CloudWatch ogs 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 CloudWatch 日志入门。
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，则必须使用较新的统一代理而不是较旧 CloudWatch 的 Logs 代理。
本节的其余部分将介绍使用较旧的 CloudWatch Logs 代理。

在正在运行的 EC2 Linux 实例上配置较旧的 CloudWatch 日志代理

您可以在现有 EC2 实例上使用 CloudWatch 日志代理安装程序来安装和配置 CloudWatch 日志代理。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。

除了使用代理之外，您还可以使用 CloudWatch 日志 SDK 或 AWS CLI日志 API 发布 CloudWatch 日志数据。最 AWS CLI 适合在命令行或通过脚本发布数据。 CloudWatch Logs SDK 最适合直接从应用程序发布日志数据或构建自己的日志发布应用程序。

步骤 1：为 CloudWatch 日志配置您的 IAM 角色或用户

CloudWatch 日志代理支持 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 日志配置您的 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 日志

安装 CloudWatch 日志代理的过程会有所不同，具体取决于您的亚马逊 EC2 实例运行的是亚马逊 Linux、Ubuntu、CentOS 还是红帽。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 日志

从 Amazon Linux AMI 2014.09 开始， CloudWatch 日志代理与 awslogs 包一起以 RPM 安装的形式提供。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 软件包安装为 RPM 而不是使用 CloudWatch 日志安装程序，您的实例可以定期接收来自的软件包更新和补丁， AWS 而无需手动重新安装 Logs 代理。 CloudWatch

> **Warning**
>
> 如果您之前使用 Python 脚本安装代理，请不要使用 RPM 安装方法更新 CloudWatch 日志代理。这样做可能会导致配置问题，从而阻止 CloudWatch 日志代理将您的日志发送到 CloudWatch。

1. 连接到 Amazon Linux 实例。有关更多信息，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 用户指南中的实例连接疑难解答。
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 日志代理参考。
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. 代理运行一段时间后，您应该会在 CloudWatch控制台中看到新创建的日志组和日志流。

   有关更多信息，请参阅 查看发送到日志的 CloudWatch 日志数据。

在现有 Ubuntu 服务器、CentOS 或红帽实例上安装和配置 CloudWatch 日志

如果您使用的是运行 Ubuntu 服务器、CentOS 或 Red Hat 的 AMI，请使用以下步骤在您的实例上手动安装 CloudWatch 日志代理。

1. 连接到您的 EC2 实例。有关更多信息，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 用户指南中的实例连接疑难解答。
2. 使用两个选项之一运行 L CloudWatch ogs 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互联网安装 Logs 代理。 CloudWatch

> **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
```

你可以通过指定 us-east-1、us-west-1、us-west-2、ap-southeas-1、ap-northeast-2、ap-southeast-1、ap-southeast-1、ap-southeast-2、ap-northeast-1、ap-northeast-1、ap-southeast-1、ap-northeast-1 来安装 CloudWatch 日志代理 1、eu-central-1、eu-west-1 或 sa-east-1 区域。

> **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。

CloudWatch Logs 代理安装程序在安装过程中需要某些信息。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
AWS 访问密钥 ID	如果使用 IAM 角色，请按 Enter。否则，请输入您的 AWS 访问密钥 ID。
AWS 秘密访问密钥	如果使用 IAM 角色，请按 Enter。否则，请输入您的 AWS 私有访问密钥。
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时间戳的格式。选择自定义可指定您自己的格式。
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 日志代理参考。

> **Note**
>
> 不支持配置多个日志源将数据发送到单个日志流。

3. 代理运行一段时间后，您应该会在 CloudWatch控制台中看到新创建的日志组和日志流。

   有关更多信息，请参阅 查看发送到日志的 CloudWatch 日志数据。

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

> **Tip**
>
> 本节中讨论的较旧 CloudWatch 的 Logs 代理即将被弃用。我们强烈建议您改用可以收集日志和指标的新统一 CloudWatch 代理。此外，较旧的 CloudWatch 日志代理需要 Python 3.3 或更早版本，默认情况下，这些版本不会安装在新的 EC2 实例上。有关统一 CloudWatch 代理的更多信息，请参阅安装代 CloudWatch 理。
> 本节的其余部分将介绍使用较旧的 CloudWatch Logs 代理。

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

您可以使用 Amazon EC2 用户数据（Amazon EC2 的一项功能，允许在启动时将参数信息传递给实例）在该实例上安装和配置 CloudWatch 日志代理。要将 CloudWatch 日志代理安装和配置信息传递给 Amazon EC2，您可以在网络位置（例如 Amazon S3 存储桶）中提供配置文件。

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 日志代理参考。

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 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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 政策。
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：

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

```
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myawsbucket/*"
            ]
        }
    ]
}
```

6. 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 Create role（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。

启动新实例并启用 CloudWatch 日志

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。
3. 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。亚马逊 Linux AMI 以及 Ubuntu 和 RHEL 的 AMI 已经包含云初始化，但是 CentOS 和其他 AMI 可能不包含云初始化。 AWS Marketplace

4. 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。
5. 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://DOC-EXAMPLE-BUCKET1/my-config-file
```

6. 对实例进行任何其他更改，检查启动设置，然后选择 Launch（启动）。
7. 代理运行一段时间后，您应该会在 CloudWatch 控制台中看到新创建的日志组和日志流。

有关更多信息，请参阅 查看发送到日志的 CloudWatch 日志数据。

快速入门：允许运行 Windows Server 2016 的 Amazon EC2 实例使用 CloudWatch 日志代理向 CloudWatch 日志发送日志

> **Tip**
>
> CloudWatch 包括一个新的统一代理，可以从 EC2 实例和本地服务器收集日志和指标。我们建议您使用更新的统一 CloudWatch代理。有关更多信息，请参阅 CloudWatch 日志入门。
> 本节的其余部分将介绍使用较旧的 CloudWatch Logs 代理。

允许运行 Windows Server 2016 的 Amazon EC2 实例使用较旧的 CloudWatch 日志代理向 CloudWatch 日志发送日志

您可以使用多种方法使运行 Windows Server 2016 的实例能够将日志发送到 CloudWatch 日志。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 发送日志、事件和性能计数器 CloudWatch。

步骤

- 下载示例配置文件
- 配置 JSON 文件用于 CloudWatch
- 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 运行命令启用 CloudWatch 日志

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`。

配置 JSON 文件用于 CloudWatch

您可以 CloudWatch 通过在配置文件中指定您的选择来确定要发送到哪些日志。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 日志
- 步骤 2：配置以下各项的设置 CloudWatch
- 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 日志

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

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

步骤 2：配置以下各项的设置 CloudWatch

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 Lo CloudWatch gs。要将相同的日志数据发送到不同的位置，您可以为每个 ID 添加具有唯一 ID（例如“CloudWatchLogs2”和 CloudWatchLogs “3”）以及不同区域的其他部分。

配置设置以将日志数据发送到 CloudWatch 日志

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 “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 “日志组” > “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。

   如果您指定的日志流名称尚不存在，Lo CloudWatch gs 会自动为您创建该名称。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 日志。

将 Windows 应用程序事件日志数据发送到 CloudWatch 日志

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 日志

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 日志

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 日志

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 的事件跟踪数据发送到 CloudWatch 日志

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 都由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，并且您的时间戳不包含时区信息，则 “ CloudWatch 日志” 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 日志

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\W3SVC`*n*）。

> **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，并且您的时间戳不包含时区信息，则 “ CloudWatch 日志” 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 日志，请(CustomLogs,ETW,SystemEventLog),CloudWatchLogs添加到该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 AWS 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 A mazon 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 运行命令配置与 CloudWatch 日志的集成之前，请验证您的实例是否满足最低要求。有关更多信息，请参阅 AWS 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 服务器实例和托管实例必须具有出站互联网访问权限才能向发送日志和事件数据 CloudWatch。有关如何配置互联网访问权限的更多信息，请参阅 Amazon VPC 用户指南中的互联网网关。

使用 Systems Manager 运行命令启用 CloudWatch 日志

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 日志的集成

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. 对于命令文档，请选择 AWS-ConfigureCloudWatch。
6. 对于目标实例，请选择要与 CloudWatch 日志集成的实例。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 S ystems 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：允许运行 Windows Server 2012 和 Windows Server 2008 的 Amazon EC2 实例向日志发送 CloudWatch 日志

> **Tip**
>
> CloudWatch 包括一个新的统一代理，可以从 EC2 实例和本地服务器收集日志和指标。我们建议您使用更新的统一 CloudWatch代理。有关更多信息，请参阅 CloudWatch 日志入门。
> 本节的其余部分将介绍使用较旧的 CloudWatch Logs 代理。

允许运行 Windows Server 2012 和 Windows Server 2008 的 Amazon EC2 实例向日志发送 CloudWatch 日志

使用以下步骤使运行 Windows Server 2012 和 Windows Server 2008 的实例能够将日志发送到 CloudWatch 日志。

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。

配置 JSON 文件用于 CloudWatch

您可以 CloudWatch 通过在 JSON 配置文件中指定您的选择来确定要发送到哪些日志。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 日志
- 步骤 2：配置以下各项的设置 CloudWatch
- 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 日志

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

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

步骤 2：配置以下各项的设置 CloudWatch

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 Lo CloudWatch gs。要将相同的日志数据发送到不同的位置，您可以为每个 ID 添加具有唯一 ID（例如 “CloudWatchLogs2” 和 CloudWatchLogs “3”）以及不同区域的其他部分。

配置设置以将日志数据发送到 CloudWatch 日志

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 “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 “日志组” > “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。

   如果您指定的日志流名称尚不存在，Lo CloudWatch gs 会自动为您创建该名称。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 日志。

将 Windows 应用程序事件日志数据发送到 CloudWatch 日志

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 日志

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 日志

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 日志

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 的事件跟踪数据发送到 CloudWatch 日志

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 都由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，并且您的时间戳不包含时区信息，则 “ CloudWatch 日志” 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 日志

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\W3SVC*n*）。

> **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，并且您的时间戳不包含时区信息，则 “ CloudWatch 日志” 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 日志，请`(CustomLogs,ETW,SystemEventLog),CloudWatchLogs`添加到该`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

要允许运行 Windows Server 2012 或 Windows Server 2008 的 Amazon EC2 实例向日志发送 CloudWatch 日志，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。有关使用早期版本的 EC2Config 的更多信息，请参阅 CloudWatch Amazon EC2 用户指南中的 “使用 EC2Config 3.x 或更早版本进行配置”

CloudWatch 使用 ec2Config 4.x 进行配置

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 服务控制面板或使用以下 PowerShell 命令启动或重新启动 SSM 代理 (`AmazonSSMAgent.exe`)：

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

SSM 代理重新启动后，它会检测配置文件并配置实例以进行集成。 CloudWatch 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。要禁用实例上的 CloudWatch 集成，请更改`IsEnabled`到配置文件`false`并将其保存在配置文件中。

快速入门：使用 AWS OpsWorks 和 Chef 安装 CloudWatch Logs 代理

您可以安装 CloudWatch 日志代理并使用第三方系统 AWS OpsWorks 和云基础架构自动化工具和 Chef 创建日志流。Chef 使用“配方”（编写的“配方”用于在计算机中安装和配置软件）和“说明书”（配方的集合）来执行其配置和策略分配任务。有关更多信息，请参阅 Chef。

下面的 Chef 配方示例显示如何在每个 EC2 实例中监控一个日志文件。配方将堆栈名称用作日志组，将实例的主机名用作日志流名称。要监控多个日志文件，您需要扩展配方，以创建多个日志组和日志流。

步骤 1：创建自定义配方

创建存储库来存储您的食谱。 AWS OpsWorks 支持 Git 和 Subversion，或者你可以在 Amazon S3 中存储档案。AWS OpsWorks 用户指南中说明书存储库部分对您的说明书存储库的结构进行了说明。以下示例假设说明书名为 `logs`。install.rb 配方用于安装 Logs 代理 CloudWatch 。你也可以下载食谱示例 (CloudWatchLogs-Cookbooks.zip)。

创建包含以下代码、名为 metadata.rb 的文件：

```
#metadata.rb

name             'logs'
version          '0.0.1'
```

创建 CloudWatch 日志配置文件：

```
#config.rb

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

下载并安装 CloudWatch 日志代理：

```
# install.rb

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

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

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

> **Note**
>
> 在上面的示例中，使用以下任一内容替换*##*：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。

如果代理安装失败，请检查以确保 python-dev 程序包已安装。如果未安装，请使用以下命令，然后重试代理安装：

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

此配方使用 cwlogs.cfg.erb 模板文件，您可以修改该文件以指定不同属性，如要对哪些文件记录日志。有关这些属性的更多信息，请参阅 CloudWatch 日志代理参考。

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


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

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

该模板通过引用堆栈配置和部署 JSON 中的相应属性获取堆栈名称与主机名。cwlogs 说明书的 default.rb 属性文件 (logs/attributes/default.rb) 定义用于指定要记录的文件的属性。

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

步骤 2：创建 AWS OpsWorks 堆栈

1. 打开 AWS OpsWorks 控制台，网址为 https://console.aws.amazon.com/opsworks/。
2. 在OpsWorks 控制面板上，选择添加堆栈以创建 AWS OpsWorks 堆栈。
3. 在 Add stack（添加堆栈）屏幕上，选择 Chef 11 stack（Chef 11 堆栈）。
4. 对于 Stack name（堆栈名称），输入一个名称。
5. 对于 Use custom Chef Cookbooks（使用自定义 Chef 说明书），选择 Yes（是）。
6. 对于 Repository type（存储库类型），选择您使用的存储库类型。如果要使用上述示例，请选择 Http Archive（Http 归档）。
7. 对于 Repository URL（存储库 URL），输入用于存储前面步骤中创建的说明书的存储库。如果要使用上述示例，请输入 **`https://s3.amazonaws.com/aws-cloudwatch/downloads/CloudWatchLogs-Cookbooks.zip`**。
8. 选择 Add Stack（添加堆栈）创建堆栈。

步骤 3：扩展您的 IAM 角色

要将 CloudWatch 日志用于您的 AWS OpsWorks 实例，您需要扩展您的实例使用的 IAM 角色。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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）页面上的 Create Your Own Policy（创建您自己的策略）下，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 政策。
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：

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

```
    ],
    "Resource": [
      "arn:aws:logs:*:*:*"
    ]
  }
 ]
}
```

6. 选择 Create Policy（创建策略）。
7. 在导航窗格中，选择角色，然后在内容窗格中的角色名称中，选择 AWS OpsWorks 堆栈使用的实例角色的名称。在堆栈设置中，可以看到堆栈使用的角色（默认角色为 `aws-opsworks-ec2-role`。）

> **Note**
>
> 选择角色名称，而不是选中复选框。

8. 在 Permissions（权限）选项卡的 Managed Policies（托管策略）下，选择 Attach Policy（附加策略）。
9. 在 Attach Policy（附加策略）页面的表标题（Filter（筛选条件）和 Search（搜索）旁边）中，选择 Policy Type（策略类型）和 Customer Managed Policies（客户托管策略）。
10. 对于Customer Managed Policies（客户托管策略），选择您在上面创建的 IAM policy，然后选择 Attach Policy（附加策略）。

    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。

步骤 4：添加层

1. 打开 AWS OpsWorks 控制台，网址为 https://console.aws.amazon.com/opsworks/。
2. 在导航窗格中，选择 Layers（层）。
3. 在内容窗格中，选择一个层，然后选择 Add layer（添加层）。
4. 在OpsWorks选项卡上，对于 “图层类型”，选择 “自定义”。
5. 对于 Name（名称）和 Short name（短名称）字段，输入层的长名称和短名称，然后选择 Add layer（添加层）。
6. 在 “食谱” 选项卡上的 “自定义厨师食谱” 下，有几个与生命周期事件相对应的标题：设置、配置、部署、取消部署和关闭。 AWS OpsWorks AWS OpsWorks 在实例生命周期中的这些关键点触发这些事件，从而运行相关的配方。

> **Note**
>
> 如果上述标题不可见，请在 Custom Chef Recipes（自定义 Chef 配方）下，选择 edit（编辑）。

7. 在 Setup（设置）旁输入 logs::config, logs::install，选择 + 将其添加到列表中，然后选择 Save（保存）。

   AWS OpsWorks 在实例启动后，立即在该层中的每个新实例上运行此配方。

步骤 5：添加实例

层仅控制如何配置实例。现在您需要将一些实例添加到层并启动它们。

1. 打开 AWS OpsWorks 控制台，网址为 https://console.aws.amazon.com/opsworks/。
2. 在导航窗格中，选择 Instances（实例），然后在您的层下选择 + Instance（+ 实例）。
3. 接受默认设置，然后选择 Add Instance（添加实例），以将该实例添加到层。
4. 在该行的 Actions（操作）列中，单击 start（启动）以启动该实例。

   AWS OpsWorks 启动新的 EC2 实例并配置 CloudWatch 日志。在实例就绪后，其状态更改为联机。

步骤 6：查看您的日志

代理运行一段时间后，您应该会在 CloudWatch 控制台中看到新创建的日志组和日志流。

有关更多信息，请参阅 查看发送到日志的 CloudWatch 日志数据。

报告 CloudWatch Logs 代理状态

使用以下过程报告您的 EC2 实例上 CloudWatch 日志代理的状态。

报告代理状态

1. 连接到您的 EC2 实例。有关更多信息，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 用户指南中的实例连接疑难解答

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. 检查 /var/log/awslogs.log 文件中是否存在任何错误、警告或 CloudWatch 日志代理问题。

启动 CloudWatch 日志代理

如果您的 EC2 实例上的 CloudWatch 日志代理在安装后没有自动启动，或者您停止了代理，则可以使用以下过程启动代理。

启动代理

1. 连接到您的 EC2 实例。有关更多信息，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 用户指南中的实例连接疑难解答。

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

使用以下过程在您的 EC2 实例上 CloudWatch 停止 Logs 代理。

停止代理

1. 连接到您的 EC2 实例。有关更多信息，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 用户指南中的实例连接疑难解答。

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
```

快速入门： AWS CloudFormation 用于开始使用 CloudWatch 日志

AWS CloudFormation 使您能够以 JSON 格式描述和配置您的 AWS 资源。这种方法的优点包括能够将一组 AWS 资源作为一个单元进行管理，并且可以轻松地跨区域复制您的 AWS 资源。

AWS 使用置备时 AWS CloudFormation，您可以创建描述要使用的 AWS 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。您可以使用设置更丰富的 CloudWatch 日志部署 AWS CloudFormation。有关模板示例的更多信息，请参阅AWS CloudFormation 用户指南中的 Amazon CloudWatch 日志模板片段。有关开始使用的更多信息，请参阅 AWS CloudFormation 用户指南中的开始使用 AWS CloudFormation。

在 AWS SDK 中使用 CloudWatch 日志

AWS 软件开发套件 (SDK) 可用于许多流行的编程语言。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以其首选语言构建应用程序。

SDK 文档	代码示例
AWS SDK for C++	AWS SDK for C++ 代码示例
AWS CLI	AWS CLI 代码示例
AWS SDK for Go	AWS SDK for Go 代码示例
AWS SDK for Java	AWS SDK for Java 代码示例
AWS SDK for JavaScript	AWS SDK for JavaScript 代码示例
AWS SDK for Kotlin	AWS SDK for Kotlin 代码示例
AWS SDK for .NET	AWS SDK for .NET 代码示例
AWS SDK for PHP	AWS SDK for PHP 代码示例
AWS Tools for PowerShell	PowerShell 代码示例工具
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 代码示例
AWS SDK for Ruby	AWS SDK for Ruby 代码示例
AWS SDK for Rust	AWS SDK for Rust 代码示例
适用于 SAP ABAP 的 AWS SDK	适用于 SAP ABAP 的 AWS SDK 代码示例
AWS SDK for Swift	AWS SDK for Swift 代码示例

有关特定于 CloudWatch 日志的示例，请参阅使用 AWS SDK 的 CloudWatch 日志的代码示例。

示例可用性

找不到所需的内容？ 通过使用此页面底部的提供反馈链接请求代码示例。

使用 “日志见解” 分析 CloudWatch 日志数据

借助 CloudWatch Logs Insights，您可以在 Amazon Logs 中以交互方式搜索和分析您的 CloudWatch 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 来识别潜在原因并验证已部署的修复程序。

CloudWatch Logs Insights 包括一种专门构建的查询语言，其中包含一些简单但功能强大的命令。CloudWatch Logs Insights 提供示例查询、命令描述、查询自动完成和日志字段发现，以帮助您入门。示例查询涉及几种类型的 AWS 服务日志。

CloudWatch Logs Insights 会自动发现 Amazon Route 53、 AWS Lambda AWS CloudTrail、和 Amazon VPC 等 AWS 服务的日志中的字段，以及任何以 JSON 形式发送日志事件的应用程序或自定义日志。

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

> **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。

您也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。这种 AI 辅助功能会根据你的提示生成查询，并 line-by-line 解释查询的工作原理。有关更多信息，请参阅使用自然语言生成和更新 L CloudWatch ogs Insights 查询。

如果您登录的账户设置为 CloudWatch 跨账户可观察性监控账户，则可以对与该监控账户关联的源账户中的 CloudWatch 日志组运行 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。

单个请求可以查询最多 50 个日志组。如果查询未在 60 分钟内完成，则会超时。查询结果在 7 天内可用。

您可以保存已创建的查询。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。

CloudWatch Logs Insights 查询会根据查询的数据量收取费用。有关更多信息，请参阅 Amazon CloudWatch 定价。

> **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 日志见解查询功能。有关更多信息，请参阅 Amazon CloudWatch 日志 API 参考StartQuery中的。

内容

- 日志类中支持的命令
- 入门：查询教程
- 支持的日志和发现的字段
- CloudWatch 日志见解查询语法
- 模式分析
- 与之前的时间范围进行比较（差异）
- 示例查询
- 在图形中可视化日志数据
- 保存并重新运行 Logs Insig CloudWatch hts 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
- 使用加密查询结果 AWS Key Management Service
- 使用自然语言生成和更新 L CloudWatch ogs Insights 查询

日志类中支持的命令

标准 CloudWatch 日志类中的日志组支持所有 Logs Insights 查询命令。Infrequent Access 日志类中的日志组支持除了`patterndiff`、和之外的所有查询命令。`unmask`

入门：查询教程

以下各节包括示例查询教程，可帮助您开始使用 L CloudWatch ogs Insights。

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 Log CloudWatch s Insights。您运行示例查询，然后查看如何修改并重新运行它。

要运行查询，您必须已经在日志中存储了 CloudWatch 日志。如果您已经在使用 CloudWatch 日志，并且已经设置了日志组和日志流，则可以开始使用了。如果您使用诸如 AWS CloudTrail Amazon Route 53 或 Amazon VPC 之类的服务，并且已将这些服务的日志设置为进入日志，则可能已经有 CloudWatch 日志。有关向 Logs 发送 CloudWatch 日志的更多信息，请参阅 CloudWatch 日志入门。

L CloudWatch ogs Insights 中的查询要么返回一组来自日志事件的字段，要么返回对日志事件执行的数学聚合或其他操作的结果。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 L CloudWatch ogs Insights 示例查询

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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（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. 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 跨账户可观察性的监控账户，则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**
   >
   > 只有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 日志显示一段时间内日志组中日志事件的条形图。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**
>
> L CloudWatch ogs Insights 中提供的一些示例查询使用`head`或`tail`命令代替`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 日志见解” 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 Logs Insig CloudWatch hts 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。要查看过去 15 分钟内 CloudWatch 日志在所选日志组中包含的日志事件中检测到的最常见字段，以及每个字段出现在这些日志事件中的百分比，请选择页面右侧的字段。

   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。L CloudWatch ogs Insights 查询中的命令必须用竖线字符分隔。
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。

CloudWatch Logs Insights 包含强大的查询功能，包括多个命令以及对正则表达式、数学和统计运算的支持。有关更多信息，请参阅 CloudWatch 日志见解查询语法。

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 L CloudWatch ogs 用户指南中支持的操作和函数。

使用聚合函数运行查询

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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 跨账户可观察性的监控账户，则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。

选择日志组时，如果日志组是标准类 CloudWatch 日志组，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。

4. 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
```

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

字段菜单位于页面的右上角，显示 Logs Insights 在您的 CloudWatch 日志组中检测到的所有已发现字段。

6. 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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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 跨账户可观察性的监控账户，则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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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 跨账户可观察性的监控账户，则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 bin(30s)
   ```

   结果显示 Logs 在每 30 秒周期内收到的日志组中 CloudWatch 日志事件的数量。
5. 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。

支持的日志和发现的字段

CloudWatch Logs Insights 支持不同的日志类型。对于发送到标准类日志组 Amazon Logs 的每份 CloudWatch 日 CloudWatch 志，Logs Insights 都会自动生成五个系统字段：

- @message 包含原始未解析的日志事件。这等同于中的message字段InputLogevent。
- @timestamp 包含日志事件的 timestamp 字段中事件时间戳。这等同于中的timestamp字段InputLogevent。
- @ingestionTime包含 L CloudWatch ogs 收到日志事件的时间。
- @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 *account-id*:*log-group-name* 形式的日志组标识符。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。

> **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 日志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。

对于带有 Logs Insights 无法自动发现的字段的其他类型的 CloudWatch 日志，您可以使用parse命令提取并创建提取的字段以用于该查询。有关更多信息，请参阅 CloudWatch 日志见解查询语法。

如果发现的日志字段的名称以@字符开头， CloudWatch 则 Logs Insights 会显示该字段，并在开头@附加一个额外的字段。例如，如果日志字段名称为 @example.com，则此字段名称显示为 @@example.com。

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

日志类型	发现的日志字段
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 会自动发现 Lambda 日志中的日志字段，但仅针对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，您可以使用点符号表示 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。有关该`parse`命令的更多信息，请参阅 Amazon CloudWatch 用户指南中的查询语法。

CloudWatch 日志见解查询语法

借助 CloudWatch Logs Insights，您可以使用查询语言来查询您的日志组。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。

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发现的字段。

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 CloudWatch 日志类中的日志组支持所有 Logs Insights 查询命令。Infrequent Access 日志类中的日志组支持除了`patterndiff`、和之外的所有查询命令。`unmask`

display	在查询结果中显示一个或多个特定字段。
fields	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。
filter	筛选查询，以仅返回与一个或多个条件匹配的日志事件。
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享文本结构。 CloudWatch Logs Insights 为您提供了分析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。
diff	将您请求的时间段内发现的日志事件与前一时间段内相同长度的日志事件进行比较，这样您就可以寻找趋势并确定某些日志事件是否是新的。
parse	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。
sort	按升序 (asc) 或降序 (desc) 顺序显示返回的日志事件。
stats	使用日志字段值计算聚合统计数据。
limit	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。
dedup	根据您指定的字段中的特定值删除重复的结果。
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。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。
其他操作和函数	CloudWatch Logs Insights 还支持许多比较、算术、日期时间、数字、字符串、IP 地址以及常规函数和操作。

以下各节提供了有关 L CloudWatch ogs Insights 查询命令的更多详细信息。

主题

- display
- fields
- 筛选条件

- 模式
- diff
- parse
- sort
- stats
- 限制
- dedup
- unmask
- 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

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。

筛选条件

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
```

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/
```

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来显示新出现的趋势、监控已知错误以及识别频繁发生或成本高昂的日志行。 CloudWatch Logs Insights 还提供了一种控制台体验，您可以使用它来查找和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。

由于该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。

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 <*> 表示。例如，[INFO] Request time: <*> 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发现的日志事件与前一时间段内相同长度的日志事件进行比较。这样，您就可以寻找趋势并确定特定的日志事件是否是新的。

在diff命令中添加一个修饰符以指定要与之比较的时间段：

- diff将当前选定时间范围内的日志事件与前一个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前一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，请参见支持的正则表达式（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 @timestamp, NetworkInterface
```

> **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
```

sort

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。

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。

- 所有非数字值都位于所有数字值之前。数字值是仅包含数字的值，不包含数字和其他字符的混合值。
- 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不同的块以进行比较。它按非数字部分的 Unicode 值排序，先按长度对数字部分进行排序，然后按其数值排序。

有关 Unicode 顺序的更多信息，请参阅 Unicode 字符列表。

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

如果按降序排序，则排序结果将相反。

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 为使用该stats函数和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 支持统计数据聚合函数和统计数据非聚合函数。

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)	number	指定的字段中值的平均值。
count() count(fieldName: LogField)	number	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，而 count(fieldName) 对包含所指定字段名称的所有记录进行计数。
count_distinct(fieldName: LogField)	number	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 count_distinct 返回的值只是一个近似值。
max(fieldName: LogField)	LogFieldValue	所查询的日志中此日志字段的值的最大值。
min(fieldName: LogField)	LogFieldValue	所查询的日志中此日志字段的值的最小值。

函数	结果类型	描述
pct(fieldName: LogFieldValue, percent: number)	LogFieldValue	百分位数指示某个值在数据集中的相对位置。例如，pct(@duration, 95) 返回 @duration 值，95% 的 @duration 值低于此值，5% 的值高于此值。
stddev(fieldName: NumericLogField)	number	指定的字段中值的标准偏差。
sum(fieldName: NumericLogField)	number	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
earliest(fieldName: LogField)	LogField	从查询的日志中具有最早时间戳的日志事件返回 fieldName 的值。
latest(fieldName: LogField)	LogField	从查询的日志中具有最晚时间戳的日志事件返回 fieldName 的值。
sortsFirst(fieldName: LogField)	LogField	返回在查询的日志中排在第一位的 fieldName 的值。
sortsLast(fieldName: LogField)	LogField	返回在查询的日志中排在最后一位的 fieldName 的值。

限制

使用 limit 指定您希望查询返回的日志事件数。

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。

布尔值、比较、数值、日期时间和其他函数

CloudWatch Logs Insights 支持查询中的许多其他操作和功能，如以下各节所述。

主题

- 算术运算符
- 布尔运算符
- 比较运算符
- 数值运算符
- 日期时间函数
- 常见函数
- 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
=	Equal
!=	Not equal
<	Less than
>	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
`abs(a: number)`	number	绝对值
`ceil(a: number)`	number	舍入到上限（大于 a 的值的最小整数）
`floor(a: number)`	number	舍入到下限（小于 a 的值的最大整数）
`greatest(a: number, ...numbers: number[])`	number	返回最大值

操作	结果类型	描述
`least(a: number, ...numbers: number[])`	number	返回最小值
`log(a: number)`	number	自然对数
`sqrt(a: number)`	number	平方根

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。使用由以下数字和其中一个组成的时间段：

- `ms`以毫秒为单位
- `s`持续几秒钟
- `m`几分钟
- `h`持续几个小时

例如，`10m` 为 10 分钟，`1h` 为 1 小时。

> **Note**
>
> 为您的日期时间函数使用最合适的时间单位。 CloudWatch 日志会根据您选择的时间单位对您的请求进行上限。例如，对于使用的所有请求，它将上限 60 作为最大值`s`。因此，如果您指定`bin(300s)`， CloudWatch Logs 实际上将其实现为 60 秒，因为 60 是一分钟内的秒数，因此 CloudWatch Logs 不会使用大于 60 的数字`s`。要创建 5 分钟的存储桶，请`bin(5m)`改用。的上限`ms`为 1000，`s`和的上限`m`为 60，的上限`h`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

`bin(period: Period)`	Timestamp	将 `@timestamp` 的值四舍五入到指定的时间段，然后截断。例如，`bin(5m)` 将 `@timestamp` 的值四舍五入到最近的 5 分钟。
		
		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。
		
		`filter @message like /Exception/`
		`| stats count(*) as exceptionCount`
		`by bin(1h)`
		`| sort exceptionCount desc`
		
		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`

函数	结果类型	描述
`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 纪元以来的毫秒数并将其转换为时间戳。
`toMillis(fieldName: Timestamp)`	number	将在命名字段中找到的时间戳转换为表示自 Unix 纪元以来毫秒数的数字。例如，`toMillis(@timestamp)` 将时间戳 `2022-01-14T13:18:031.000-08:00` 转换为 `1642195111000` 。

> **Note**
>
> 目前， CloudWatch Logs Insights 不支持筛选带有人类可读时间戳的日志。

常见函数

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
`ispresent(fieldName: LogField)`	布尔值	如果字段存在，则返回 `true`
`coalesce(fieldName: LogField, ...fieldNames: LogField[])`	LogField	返回列表中的第一个非 null 值

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
isValidIp(fieldName: string)	布尔值	如果字段是有效的 IPv4 或 IPv6 地址，则返回 true。
isValidIpV4(fieldName: string)	布尔值	如果字段是有效的 IPv4 地址，则返回 true。
isValidIpV6(fieldName: string)	布尔值	如果字段是有效的 IPv6 地址，则返回 true。
isIpInSubnet(fieldName: string, subnet: string)	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 true。指定子网时，请使用 CIDR 表示法（例如 192.0.2.0/24 或 2001:db8::/32 ），其中 192.0.2.0 或 2001:db8:: 是 CIDR 块的起点。
isIpv4InSubnet(fieldName: string, subnet: string)	布尔值	如果字段是指定的 v4 子网中有效的 IPv4 地址，则返回 true。指定子网时，请使用 CIDR 表示法（例如 192.0.2.0/24 ），其中 192.0.2.0 是 CIDR 块的起点。
isIpv6InSubnet(fieldName: string, subnet: string)	布尔值	如果字段是指定的 v6 子网中有效的 IPv6 地址，则返回 true。指定子网时，请使用 CIDR 表示法（例如 2001:db8::/32 ），其中 2001:db8:: 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
isempty(fieldName: string)	数字	如果字段缺失或为空字符串，则返回 1。
isblank(fieldName: string)	数字	如果字段缺失或为空字符串，或只包含空格，则返回 1。
concat(str: string, ...strings: string[])	字符串	连结字符串。
ltrim(str: string) ltrim(str: string, trimChars: string)	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 str 左侧删除 trimChars 中的字符。例如，ltrim("xyZxyfooxyZ","xyZ") 将返回 "fooxyZ"。
rtrim(str: string) rtrim(str: string, trimChars: string)	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 str 右侧删除 trimChars 字符。例如，rtrim("xyZfooxyxyZ","xyZ") 将返回 "xyZfoo"。
trim(str: string) trim(str: string, trimChars: string)	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 str 两端删除 trimChars 字符。例如，trim("xyZ

函数	结果类型	描述
		xyfooxyxyZ","xyZ") 将返回 "foo"。
strlen(str: string)	number	返回 Unicode 代码点中字符串的长度。
toupper(str: string)	字符串	将字符串转换为大写。
tolower(str: string)	字符串	将字符串转换为小写。
substr(str: string, startIndex: number) substr(str: string, startIndex: number, length: number)	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，substr("xyZfooxyZ",3, 3) 将返回 "foo"。
replace(fieldName: string, searchValue: string, replaceValue: string)	字符串	将 fieldName: string 中出现的所有 searchValue 替换为 replaceValue 。 例如，函数 replace(logGroup,"smoke_test","Smoke") 搜索录入事件，其中字段 logGroup 包含字符串值 smoke_test ，并使用字符串 Smoke 替换该值。
strcontains(str: string, searchValue: string)	number	如果 str 包含 searchValue ，则返回 1，否则返回 0。

包含特殊字符的字段

如果字段包含@符号或句点 () 以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 日志见解支持查询中的评论。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
```

模式分析

CloudWatch 当您查询日志时，Logs Insights 使用机器学习算法来查找模式。模式是一种共享的文本结构，在您的日志字段中重复出现。查看查询结果时，可以选择 “模式” 选项卡，查看 CloudWatch Logs 根据结果样本找到的模式。或者，您可以将`pattern`命令附加到查询中，以分析整组匹配的日志事件中的模式。

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以压缩成几个模式。

考虑以下三个日志事件的示例。

```
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
```

在前面的示例中，所有三个日志事件都遵循一种模式：

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

模式中的字段称为标记。模式中不同的字段（例如请求 ID 或时间戳）是动态标记。每个动态令牌都由 Lo CloudWatch gs 显示`<*>`时表示。

动态令牌的常见示例包括错误代码、时间戳和请求 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，则标记值可能是`501`。

模式检测还用于 CloudWatch 日志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）。

模式分析入门

模式检测将在任何 Logs Insight CloudWatch s 查询中自动执行。不包含该`pattern`命令的查询会在结果中同时获取日志事件和模式。

如果在查询中包含该`pattern`命令，则会对整个匹配的日志事件集执行模式分析。这为您提供了更准确的模式结果，但是当您使用该`pattern`命令时，不会返回原始日志事件。当查询不包含时`pattern`，模式结果要么基于返回的前 1000 个日志事件，要么基于您在查询中使用的限制值。如果包含`pattern`在查询中，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。

要开始在 Log CloudWatch s Insights 中进行模式分析

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择日志、日志、见解。

   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，使查询如下所示：

   ```
   fields @timestamp, @message, @logStream, @log
   | sort @timestamp desc
   ```
4. 在选择日志组下拉列表中，选择一个或多个要查询的日志组。
5. （可选）使用时间间隔选择器选择要查询的时间段。

   您可以在 5 分钟和 30 分钟间隔、1 小时、3 小时和 12 小时间隔或自定义时间范围之间进行选择。
6. 选择 “运行查询” 以开始查询。

   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。表格上方有一条关于有多少条记录与查询相匹配的消息，类似于显示 71,101 条匹配记录中的 1000 条。
7. 选择 “图案” 选项卡。
8. 现在，该表显示了在查询中找到的模式。由于查询中不包含该`pattern`命令，因此此选项卡仅显示在 “日志” 选项卡的表中显示的 1000 个日志事件中发现的模式。

   对于每种模式，都会显示以下信息：

   - 模式，每个动态令牌都显示为`<*>`。

- 事件计数，即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以按频率对模式进行排序。
- 事件比率，即包含此模式的查询日志事件的百分比。
- 严重性类型，将是以下类型之一：
  - 如果模式中包含 “错误” 一词，则出错。
  - 如果模式包含 “警告” 一词但不包含错误，则发出警告。
  - 如果模式不包含 “警告” 或 “错误”，则为@@ 信息。

  选择严重性信息列标题以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`，这样完整的查询如下所示：

```
fields @timestamp, @message, @logStream, @log
| pattern @message
```

10. 选择运行查询。

    查询完成后，“日志” 选项卡中没有任何结果。但是，根据查询的日志事件总数，“模式” 选项卡列出的模式数量可能更多。

11. 无论您是否包含`pattern`在查询中，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：

    - 图案。在模式中选择一个代币来分析该代币的值。
    - 显示查询时间范围内该模式出现次数的直方图。这可以帮助您识别有趣的趋势，例如模式的发生率突然增加。
    - 日志样本选项卡显示了一些与所选模式相匹配的日志事件。
    - 标记值选项卡会显示所选动态令牌的值（如果您选择了动态标记）。

    > **Note**
    >
    > 每个令牌最多捕获 10 个令牌值。代币数量可能不精确。 CloudWatch 日志使用概率计数器来生成代币数量，而不是绝对值。

- “相关模式” 选项卡显示经常出现的其他模式，这些模式与您正在检查的模式几乎同时发生。例如，如果一条ERROR消息的模式通常伴随着另一个标记为其他详细信息的日志事件，则此处会显示该模式。INFO

有关模式命令的详细信息

本节包含有关该pattern命令及其用途的更多详细信息。

- 在上一教程中，我们在添加sort命令时删除了该命令，pattern因为如果查询在pattern命令后面包含命令，则查询无效。sort在 a pattern 之前加一个是有效的sort。

  有关pattern语法的更多详细信息，请参阅模式。
- 在查询pattern中使用时，@message必须是pattern命令中选定的字段之一。
- 可以在filter命令前添加该pattern命令，以便仅将经过筛选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。

与之前的时间范围进行比较（差异）

您可以使用 CloudWatch Logs Insights 来比较日志事件在一段时间内的变化。您可以将最近一段时间内提取的日志事件与前一时间段的日志进行比较。或者，您可以与过去的类似时间段进行比较。这可以帮助您确定日志中的错误是最近引入的还是已经发生的，还可以帮助您发现其他趋势。

比较查询仅返回结果中的模式，不返回原始日志事件。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，您可以查看您感兴趣的模式的原始日志事件示例。有关日志模式的更多信息，请参阅模式分析。

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段和比较时段。比较周期的长度始终与您的原始查询周期相等。比较的默认时间间隔如下。

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

> **Note**
>
> 使用比较的查询会产生类似于在合并时间范围内运行单个 CloudWatch Logs Insights 查询的费用。有关更多信息，请参阅 Amazon CloudWatch 定价。

运行比较查询

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择日志、日志、见解。

   默认查询出现在查询框中。
3. 保留默认查询或输入其他查询。
4. 在选择日志组下拉列表中，选择一个或多个要查询的日志组。
5. （可选）使用时间间隔选择器选择要查询的时间段。默认查询是针对前一小时的日志数据。
6. 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，然后选择 Apply。
7. 选择运行查询。

   为了使查询从比较周期中获取数据，该`diff`命令会附加到您的查询中。
8. 选择 “模式” 选项卡以查看结果。

   该表显示以下信息：

   - 每个模式，模式的可变部分都被动态标记符号所取代`<*>`。有关更多信息，请参阅 模式分析。
   - 事件计数是在原始、更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内匹配日志事件的数量与比较时段内的匹配日志事件数量之间的差异。积极的不同意味着当前时间段内会有更多此类事件。
   - 差异描述简要概述了当前时间段和比较期之间该模式的变化。
   - 严重性类型是使用此模式的日志事件的可能严重性，基于日志事件中发现的单词`FATAL`，例如`ERROR`、和`WARN`。
9. 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：

   - 图案。在模式中选择一个代币来分析该代币的值。

- 显示查询时间范围内该模式出现次数的直方图。这可以帮助您识别有趣的趋势，例如模式的发生率突然增加。
- 日志样本选项卡显示了一些与所选模式相匹配的日志事件。
- 标记值选项卡会显示所选动态令牌的值（如果您选择了动态标记）。

> **Note**
>
> 每个令牌最多捕获 10 个令牌值。代币数量可能不精确。 CloudWatch 日志使用概率计数器来生成代币数量，而不是绝对值。

- “相关模式” 选项卡显示经常出现的其他模式，这些模式与您正在检查的模式几乎同时发生。例如，如果一条ERROR消息的模式通常伴随着另一个标记为其他详细信息的日志事件，则此处会显示该模式。INFO

示例查询

本节包含可在CloudWatch 控制台中运行的一般和有用的查询命令列表。有关如何运行查询命令的信息，请参阅 Amazon L CloudWatch ogs 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 日志见解查询语法。

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
- 查询日 CloudTrail 志
- 查询 Amazon API Gateway
- NAT 网关的查询
- 查询 Apache 服务器日志
- 针对亚马逊的查询 EventBridge
- 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 provisonedMemoryMB,
        min(@maxMemoryUsed / 1000 / 1000) as smallestMemoryRequestMB,
        avg(@maxMemoryUsed / 1000 / 1000) as avgMemoryUsedMB,
        max(@maxMemoryUsed / 1000 / 1000) as maxMemoryUsedMB,
        provisonedMemoryMB - 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
```

查询日 CloudTrail 志

查找每项服务、事件类型和 AWS 区域的日志条目数。

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

查找在给定 AWS 区域中启动或停止的 Amazon EC2 主机。

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

查找新创建的 IAM 用户的 AWS 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
```

查询 Amazon API Gateway

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
```

为您的 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

如果您发现 AWS 账单中的费用高于正常水平，则可以使用 L CloudWatch ogs Insights 来查找排名靠前的贡献者。有关以下查询命令的更多信息，请参阅如何在 VPC 中找到通过 NAT 网关的流量的最大贡献者？ 在 AWS 高级支持页面上。

> **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 AWS 云运营与迁移博客上的使用 CloudWatch 日志见解简化 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
```

针对亚马逊的查询 EventBridge

获取按 EventBridge 事件详情类型分组的事件数量

```
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
```

在图形中可视化日志数据

您可以使用条形图、折线图和堆积面积图等可视化来更有效地识别日志数据中的模式。 CloudWatch Logs Insights 为使用该`stats`函数和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。

保存并重新运行 Logs Insig CloudWatch hts 查询

创建查询后，可以将其保存，以便稍后再次运行查询。查询以文件夹结构保存，因此您可以对其进行整理。每个账户每个区域可以保存多达 1000 个查询。

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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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。

   如果看不到“保存”按钮，则需要更改为 L CloudWatch ogs 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PutQueryDefinition。

运行保存的查询

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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

运行查询后，您可以将查询添加到 CloudWatch 仪表板或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。这些查询计入您的并发 L CloudWatch ogs Insights 查询上限（30 个）。

将查询结果添加到控制面板

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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。每个账户只能同时进行 30 个 CloudWatch Logs Insights 查询，包括添加到仪表板的查询。

查看您的最近查询历史记录

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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 日志控制台的新设计，请选择 “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 “状态” 下，“ CloudWatch 日志” 显示当前正在运行的所有查询的进行中。

使用加密查询结果 AWS Key Management Service

默认情况下， CloudWatch Logs 使用默认 CloudWatch 的 Logs 服务器端加密方法对 CloudWatch Logs Insights 查询的存储结果进行加密。您可以选择使用 AWS KMS 密钥来加密这些结果。如果您将 AWS KMS 密钥与加密结果相关联，则 CloudWatch Logs 将使用该密钥对账户中所有查询的存储结果进行加密。

如果您稍后解除密钥与查询结果的关联，则 CloudWatch 日志会回到默认的加密方法以供日后查询使用。但是，在关联密钥时运行的查询仍使用该密钥进行加密。 CloudWatch 解除关联 KMS 密钥后，日志仍然可以返回这些结果，因为 CloudWatch 日志仍然可以继续引用该密钥。但是，如果密钥稍后被禁用，则 CloudWatch Logs 将无法读取使用该密钥加密的查询结果。

> **Important**
>
> CloudWatch 日志仅支持对称 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。
- 如果您撤消 CloudWatch 日志对关联密钥的访问权限或删除关联的 KMS 密钥，则无法再检索 CloudWatch 日志中的加密数据。
- 您不能使用 CloudWatch 控制台关联密钥，必须使用 AWS CLI 或日 CloudWatch 志 API。

步骤 1：创建一个 AWS 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。通过此步骤，您将授予 CloudWatch 日志服务主体使用密钥的权限。该服务主体必须位于存储密钥的同一 AWS 区域。

作为最佳实践，我们建议您将密钥的使用限制在您指定的 AWS 账户中。

首先，`policy.json`使用以下get-key-policy命令保存 KMS 密钥的默认策略：

```
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章节来增强 AWS KMS 密钥的安全性。有关更多信息，请参阅 AWS KMS 密钥和加密上下文。

此示例中的Condition部分将 AWS KMS 密钥的使用限制为指定账户中的 CloudWatch Logs Insights 查询结果。

```
{
 "Version": "2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::account_ID:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.region.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt*",
                "kms:Decrypt*",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:Describe*"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:logs:region:account_ID:query-result:*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "Your_account_ID"
                }
            }
        }
    ]
```

```
}
```

最后，使用以下put-key-policy命令添加更新的策略：

```
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

按以下方式使用 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

要取消与查询结果关联的 KMS 密钥的关联，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:*"
```

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

> **Note**
>
> 该功能通常在美国东部（弗吉尼亚北部）、美国西部（俄勒冈）和亚太地区（东京）提供 CloudWatch Logs。

CloudWatch Logs 支持自然语言查询功能，可帮助您生成和更新 L CloudWatch ogs Insights 和 M CloudWatch etrics Insigh t s 的查询。

借助此功能，您可以用通俗易懂的英语提问或描述您要查找的 CloudWatch 日志数据。自然语言功能会根据您输入的提示生成查询，并 line-by-line 解释查询的工作原理。您也可以更新查询以进一步调查数据。

根据您的环境，您可以输入诸如 “按传输字节数计算的前 100 个源 IP 地址是什么？” 之类的提示 和 “查找最慢的 10 个 Lambda 函数请求”。

要生成具有此功能的 CloudWatch Logs Insights 查询，请打开 Lo CloudWatch gs 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，必须使用CloudWatchLogsFullAccess、CloudWatchLogsReadOnlyAccessAdministratorAccess、或ReadOnlyAccess策略。
> 您也可以将 `cloudwatch:GenerateQuery` 操作包含在新的或现有的客户托管策略或内联策略中。

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。此示例显示了一个执行基本搜索的查询。

提示

以下是一个提示示例，它指示功能搜索最慢的 10 个 Lambda 函数调用。

```
Find the 10 slowest requests
```

查询

以下是自然语言功能根据提示生成的查询示例。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。

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。现在，该提示不再显示搜索最慢的 10 个 Lambda 函数调用的提示，而是引导用户搜索最慢的 20 个 Lambda 函数调用，并在另一列中添加其他日志事件。

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

更新版查询

以下是更新版查询的示例：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 L CloudWatch ogs 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 AWS Service Terms 和 AWS 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。要选择退出所有 Lo CloudWatch gs AI 功能（包括查询生成功能）的数据收集，您必须为 Logs 创建退出策略。 CloudWatch 有关更多信息，请参阅《AWS Organizations 用户指南》中的 AI 服务选择退出策略。

日志异常检测

您可以为每个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，并在日志数据中查找异常。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。

为日志组创建异常检测器后，它会使用日志组中过去两周的日志事件进行训练，进行训练。训练时间最长可能需要 15 分钟。训练完成后，它开始分析传入的日志以识别异常，异常将显示在 CloudWatch 日志控制台中供您检查。

CloudWatch 日志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以压缩成几个模式。

例如，请参阅以下三个日志事件的示例。

```
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
```

在前面的示例中，所有三个日志事件都遵循一种模式：

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

模式中的字段称为标记。模式中不同的字段（例如请求 ID 或时间戳）被称为动态标记。动态令牌由 CloudWatch Logs <*> 何时显示模式表示。为动态令牌找到的每个不同值都称为令牌值。

动态令牌的常见示例包括错误代码、时间戳和请求 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，将根据已知趋势评估日志。异常检测器将重大波动标记为异常。

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被分配一个优先级。找到的每种模式都被分配了严重性。

- 优先级是自动计算的，它基于模式的严重性级别和与预期值的偏差量。例如，如果某个令牌值突然增加 500%，则即使该异常的严重性为，也可能被指定为HIGH优先级。NONE
- 严重性仅基于模式中的关键字FATAL，例如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。

异常可见时间

创建异常检测器时，需要为其指定最长异常可见期。这是异常在控制台中显示并由 ListAnomaliesAPI 操作返回的天数。异常经过这段时间后，如果异常继续发生，则该异常会被自动接受为常规行为，异常检测器模型将停止将其标记为异常。

如果您在创建异常检测器时不调整可见时间，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该异常。抑制异常会使异常检测器在您指定的时间内停止将此事件标记为异常。隐藏异常时，可以选择仅隐藏该特定异常，或者隐藏与发现该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

是否 AWS 使用我的数据来训练机器学习算法以供其他客户 AWS 使用或供其他客户使用？

不是。 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 AWS 账户中使用。

哪些类型的日志事件适用于异常检测？

日志异常检测非常适合：应用程序日志和其他类型的日志，其中大多数日志条目都符合典型模式。包含日志级别或严重性关键字（例如 INFO、ERROR 和 DEBUG）的事件的日志组特别适合记录异常检测。

日志异常检测不适合：记录具有极长 JSON 结构的事件，例如 CloudTrail 日志。模式分析最多只能分析日志行的前 1500 个字符，因此任何超出该限制的字符都将被跳过。

审计或访问日志（例如 VPC 流日志）在异常检测方面的成功率也会降低。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。

为了帮助您确定异常检测器是否适用于某个日志组，请使用 CloudWatch 日志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 个左右，则异常检测可能效果很好。有关模式分析的更多信息，请参见模式分析。

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：

- 一种日志事件，其模式在日志组中以前从未见过。
- 与已知模式的显著差异。
- 动态代币的新值，具有一组离散的常用值。
- 动态令牌的值出现次数变化很大。

虽然前面的所有项目都可能被标记为异常，但它们并不都意味着应用程序的性能很差。例如， higher-than-usual许多200成功值可能会被标记为异常。在这种情况下，你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怎样？

不会对被屏蔽为敏感数据的日志事件的任何部分进行异常扫描。有关屏蔽敏感数据的更多信息，请参阅使用屏蔽帮助保护敏感日志数据。

在日志组上启用异常检测

使用以下步骤使用 CloudWatch 控制台创建日志异常检测器，用于扫描日志组中是否存在异常。

您也可以通过编程方式创建异常探测器。有关更多信息，请参阅CreateLogAnomalyDetector。

创建日志异常检测器

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 选择 “日志”、“记录异常”。
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. 在异常探测器名称中输入探测器的名称。
6. （可选）将评估频率从默认的 5 分钟更改为其他值。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 分钟批量接收一次新的日志事件，则将评估频率设置为 15 分钟可能是合适的。
7. （可选）要将异常检测器配置为仅在包含某些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 “异常检测过滤器模式” 中输入一个模式。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）要测试您的过滤器模式，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）要将异常可见性周期更改为默认值或将 AWS KMS 密钥与此异常检测器相关联，请选择高级配置。

   a. 要更改异常可见性周期的默认值，请在最长异常可见性周期（天）中输入一个新值。

   b. 要将 AWS KMS 密钥与此异常检测器关联，请在 KMS 密钥 ARN 中输入 ARN。如果您分配了密钥，则该探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，也必须拥有异常检测器的权限，才能检索有关其发现的异常的信息。

      您还必须确保 CloudWatch 日志服务主体有权使用该密钥。有关更多信息，请参阅 使用对异常检测器及其结果进行加密 AWS KMS。

9. 选择 “启用异常检测”。

   异常检测器已创建完毕，并根据日志组正在接收的日志事件开始训练其模型。大约 15 分钟后，异常检测处于活动状态，并开始发现和显示异常。

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 控制台查看他们发现的异常。

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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 选择 “日志”、“记录异常”。

   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日志异常。表中的每一行都显示以下信息：

   - 异常列显示异常的简短摘要。这些摘要由 CloudWatch 日志生成。
   - 异常的优先级。优先级是根据日志事件的变化量、关键词（例如在日志事件中`Exception`发生的）等来自动计算的。
   - 异常所基于的日志模式。有关模式的更多信息，请参阅日志异常检测。

- 异常日志趋势显示直方图，描绘与该模式匹配的日志量。
- 上次检测时间显示最近发现此异常的时间。
- 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，请选择其所在行的单选按钮。

   将出现 “模式检查” 窗格并显示以下内容：

   - 此异常所基于的模式。在模式中选择一个代币来分析该代币的值。
   - 显示查询时间范围内异常出现次数的直方图。
   - 日志样本选项卡显示了作为异常一部分的一些日志事件。
   - 标记值选项卡会显示所选动态令牌的值（如果您选择了动态标记）。

   > **Note**
   >
   > 每个令牌最多捕获 10 个令牌值。代币数量可能不精确。 CloudWatch 日志使用概率计数器来生成代币数量，而不是绝对值。

4. 要抑制异常，请选择其所在行中的单选按钮，然后执行以下操作：

   a. 选择操作，抑制异常。

   b. 然后指定要抑制异常的时间。

   c. 要隐藏与该模式相关的所有异常，请选择隐藏模式式。

   d. 选择 “抑制异常”。

查看在单个日志组中发现的异常情况

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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日志异常。表中的每一行都显示以下信息：

   - 异常列显示异常的简短摘要。这些摘要由 CloudWatch 日志生成。

- 异常的优先级。优先级是根据日志事件的变化量、关键词（例如在日志事件中Exception发生的）等来自动计算的。
- 异常所基于的日志模式。有关模式的更多信息，请参阅日志异常检测。
- 异常日志趋势显示直方图，描绘与该模式匹配的日志量。
- 上次检测时间显示最近发现此异常的时间。
- 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，请选择其所在行的单选按钮。

   将出现 “模式检查” 窗格并显示以下内容：

   - 此异常所基于的模式。在模式中选择一个代币来分析该代币的值。
   - 显示查询时间范围内异常出现次数的直方图。
   - 日志样本选项卡显示了作为异常一部分的一些日志事件。
   - 标记值选项卡会显示所选动态令牌的值（如果您选择了动态标记）。

   > **Note**
   >
   > 每个令牌最多捕获 10 个令牌值。代币数量可能不精确。 CloudWatch 日志使用概率计数器来生成代币数量，而不是绝对值。

5. 要抑制异常，请选择其所在行中的单选按钮，然后执行以下操作：

   a. 选择操作，抑制异常。
   b. 然后指定要抑制异常的时间。
   c. 要隐藏与该模式相关的所有异常，请选择隐藏模式。
   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。您可以指定警报在指定时间段内在日志组中发现指定数量的异常时进入ALARM状态。您还可以使用过滤器，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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择日志，日志异常。

将显示日志异常检测器表。

3. 选择要为其设置警报的异常检测器的单选按钮，然后选择创建警报。

   CloudWatch 警报创建向导出现。该LogAnomalyDetector字段显示您选择的异常检测器的名称。将显示 “指标名称” 字段AnomalyCount。

4. （可选）要筛选此警报的异常优先级，请执行以下操作之一：

   - 要使警报仅计数高优先级异常，请输入 for。**HIGH** LogAnomalyPriority
   - 要使警报仅计数高优先级和中等优先级的异常，请输入 for。**MEDIUM** LogAnomalyPriority

   有关优先级的更多信息，请参阅异常和模式的严重性和优先级。

5. 选择为警报使用静态或指标异常检测阈值。此选择决定了警报阈值的设置方式。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它 CloudWatch 决定了常用值的范围，如果实际计数超过该波段的阈值，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. For Wh ***your-metric-name***enever is。。 ，选择 “更大”、“大于/等于”、“较低/ 等于” 或 “更低”。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 Period 指定的时间内发现的警报数量超过此数量，则警报将进入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，请参阅评估警报。

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 警报如何处理丢失的数据。

9. 选择下一步。

10. 在 “通知” 中，选择 “添加通知”，然后指定警报转换为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来源的 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）如果您希望警报对 Amazon EC2 Auto Scaling、Amazon EC2、票证或 AWS Systems Manager执行操作，请选择相应的按钮，然后指定警报状态和操作。

> **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，请参阅配置 CloudWatch 为创建 OpsItems和事件创建。

12. 选择下一步。
13. 在 Name and description（名称和描述）下，输入告警的名称和描述，然后选择 Next（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，该格式仅显示在 CloudWatch 控制台的警报详细信息选项卡中。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 Preview and create（预览和创建）下，确认告警的信息和条件正确，然后选择 Create alarm（创建告警）。

日志异常检测器发布的指标

CloudWatch 日志将AnomalyCount指标发布到 CloudWatch 指标。此指标已发布到AWS/Logs命名空间。

该AnomalyCount指标发布时采用以下维度：

- LogAnomalyDetector— 异常探测器的名称
- LogAnomalyPriority— 异常的优先级

使用对异常检测器及其结果进行加密 AWS KMS

异常检测器数据始终在 CloudWatch 日志中加密。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 AWS Key Management Service 进行这一加密。如果这样做，则使用密 AWS KMS 钥进行加密。通过将 K AWS KMS MS 密钥与异常检测器关联，可以在异常检测器级别启用加密。

> **Important**
>
> CloudWatch 日志仅支持对称 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。
- 如果您撤消 CloudWatch 日志对关联密钥的访问权限或删除关联的 KMS 密钥，则无法再检索 CloudWatch 日志中的加密数据。

步骤 1：创建 AWS 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 AWS 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 日志服务主体使用密钥的权限。此服务主体必须位于存储 KMS 密钥的同一 AWS 区域。

作为最佳实践，我们建议您将 KMS 密钥的使用仅限于您指定的 AWS 账户或异常检测器。

首先，`policy.json`使用以下get-key-policy命令保存 KMS 密钥的默认策略：

```
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`章节来增强 AWS KMS 密钥的安全性。有关更多信息，请参阅 AWS KMS 密钥和加密上下文。

本示例中的`Condition`部分将 AWS KMS 密钥的使用限制为指定的账户，但它可以用于任何异常检测器。

```
{
  "Version": "2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::Your_account_ID:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
```

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

最后，使用以下put-key-policy命令添加更新的策略：

```
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

在控制台中或使用 AWS CLI 或 API 创建 KMS 密钥时，可以将其与异常检测器关联。

步骤 4：取消密钥与异常检测器的关联

将密钥与异常检测器关联后，您将无法更新该密钥。移除密钥的唯一方法是删除异常检测器，然后重新创建它。

使用日志组和日志流

日志流是共享同一来源的一系列日志事件。Logs 中每个单独的 CloudWatch 日志源构成一个单独的日志流。

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。

使用本部分中的过程处理日志组和日志流。

在 Log CloudWatch s 中创建日志组

当您按照《Amazon CloudWatch 日志用户指南》前几节中的步骤在 Amazon CloudWatch EC2 实例上安装日志代理时，将在该过程中创建日志组。您也可以直接在 CloudWatch 控制台中创建日志组。

创建日志组

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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 日志会自动接收来自多个 AWS 服务的日志事件。您也可以使用以下方法之一将其他 CloudWatch 日志事件发送到 Logs：

- CloudWatch 代理 — 统一 CloudWatch 代理可以将指标和日志发送到 CloudWatch 日志。有关安装和使用 CloudWatch 代理的信息，请参阅《亚马逊 CloudWatch 用户指南》中的使用 CloudWatch 代理从 Amazon EC2 实例和本地服务器收集指标和日志。

- AWS CLI— 将成批的日志事件put-log-events上传到 CloudWatch 日志。
- 以编程方式 — PutLogEventsAPI 使您能够以编程方式将成批的日志事件上传到 CloudWatch 日志。

查看发送到日志的 CloudWatch 日志数据

您可以 stream-by-stream 根据日志代理发送到 Logs 的日志数据来 CloudWatch 查看和滚动浏览 CloudWatch 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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择 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 Live Tail 近乎实时地查看日志

CloudWatch Logs Live Tail 可在接收新日志事件时查看事件的流式列表，从而帮助您快速排除故障。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 Pric ing 中的“日志”选项卡。

> **Note**
>
> 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。

以下各节说明了如何在控制台中使用 Live Tail。您也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 SD AWS K 启动 Live Tail 会话。

开始 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 日志见解”页面启动 Live Tail 会话。

> **Note**
>
> 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。

开始 Live Tail 会话

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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。您可以在此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何日志事件的左侧；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。
   - 为了防止会话因为许多事件与过滤器匹配而滚动得太快， CloudWatch 日志可能只显示一些匹配的事件。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)。

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， AWS CLI 也可以使用搜索特定的日志流。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。

如果您登录的账户设置为 CloudWatch 跨账户可观察性监控账户，则可以搜索和筛选与该监控账户关联的源账户中的日志事件。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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择 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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择 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。

使用搜索日志条目 AWS CLI

您可以使用搜索符合指定条件的日志条目 AWS CLI。

要使用搜索日志条目 AWS CLI

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]
```

要使用搜索给定时间范围内的日志条目 AWS CLI

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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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，请从列表中选择一个。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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。

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，请从列表中选择一个。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：

- 如果您使用的是 AWS CLI，则可以将搜索范围限制为仅搜索您感兴趣的日志流。例如，如果您的日志组有 1000 个日志流，但您只想查看三个已知相关的日志流，则可以使用将搜索限制 AWS CLI 为仅搜索日志组中的这三个日志流。
- 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。

更改日志中的 CloudWatch 日志数据保留期

默认情况下，日志数据无限期地存储在 CloudWatch 日志中。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**
>
> CloudWatch Logs 不会在日志事件达到保留设置时立即将其删除。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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，依次选择日志和日志组。
3. 找到要更新的日志组。
4. 在该日志组的 “保留” 列中，选择当前的保留期设置，例如 “永不过期”。
5. 在 “保留设置” 中，对于 “事件过期”，选择日志保留值，然后选择 “保存”。

在 Amazon 日志中标记 CloudWatch 日志组

您可以将自己的元数据以标签的形式分配给您在 Amazon Logs 中创建的 CloudWatch 日志组。标签是您为日志组定义的键值对。使用标签是一种管理 AWS 资源和整理数据（包括账单数据）的简单而强大的方法。

> **Note**
>
> 您可以使用标签来控制对 CloudWatch 日志资源的访问权限，包括日志组和目标。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。

内容

- 有关标签的基本知识
- 使用标签跟踪成本
- 标签限制
- 使用标记日志组 AWS CLI
- 使用日志 API 标记 CloudWatch 日志组

有关标签的基本知识

您可以使用或 AWS CloudFormation AWS CLI日 CloudWatch 志 API 来完成以下任务：

- 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 AWS 费用进行分类和跟踪。当您将标签应用于 AWS 资源（包括日志组）时，您的 AWS 成本分配报告包括按标签汇总的使用量和成本。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 AWS 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。
- 标签密钥不能以开头，`aws:`因为此前缀已保留供使用 AWS。 AWS 代表您创建以此前缀开头的标签，但您无法对其进行编辑或删除。

- 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：`_ . / = + - @`。

使用标记日志组 AWS CLI

您可以使用 AWS 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

为指定 CloudWatch 的 Logs 资源分配一个或多个标签（键值对）。

list-tags-for-resource

显示与 CloudWatch 日志资源关联的标签。

untag-resource

从指定的 L CloudWatch ogs 资源中移除一个或多个标签。

使用日志 API 标记 CloudWatch 日志组

您可以使用 CloudWatch 日志 API 添加、列出和移除标签。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

为指定 CloudWatch 的 Logs 资源分配一个或多个标签（键值对）。

ListTagsForResource

显示与 CloudWatch 日志资源关联的标签。

UntagResource

从指定的 L CloudWatch ogs 资源中移除一个或多个标签。

使用加密日志中的 CloudWatch 日志数据 AWS Key Management Service

日志组数据始终在 CloudWatch 日志中加密。默认情况下， CloudWatch Logs 对静态日志数据使用服务器端加密。此外，您也可以使用 AWS Key Management Service 进行这一加密。如果这样做，则使用密 AWS KMS 钥进行加密。在日志组级别启用加密，方法 AWS KMS 是在创建日志组时或日志组存在之后，将 KMS 密钥与日志组关联起来。

> **Important**
>
> CloudWatch 日志现在支持加密上下文，使用`kms:EncryptionContext:aws:logs:arn`作为密钥，使用日志组的 ARN 作为该密钥的值。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 AWS KMS 密钥和加密上下文。

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。这些数据在整个保留期内以加密格式存储。 CloudWatch 只要有请求，日志就会解密这些数据。 CloudWatch 每当请求加密数据时，日志都必须具有 KMS 密钥的权限。

如果您稍后解除 KMS 密钥与日志组的关联，Logs 将使用 CloudWatch CloudWatch 日志默认加密方法对新摄取的数据进行加密。之前采集的所有使用 KMS 密钥加密的数据仍使用 KMS 密钥进行加密。CloudWatch 解除关联 KMS 密钥后，日志仍然可以返回该数据，因为 CloudWatch 日志仍然可以继续引用该密钥。但是，如果密钥稍后被禁用，则 CloudWatch Logs 将无法读取使用该密钥加密的日志。

> **Important**
>
> CloudWatch 日志仅支持对称 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。
- 如果您撤消 CloudWatch 日志对关联密钥的访问权限或删除关联的 KMS 密钥，则无法再检索 CloudWatch 日志中的加密数据。
- 您无法使用 CloudWatch 控制台将 KMS 密钥与日志组关联。

步骤 1：创建 AWS 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 AWS 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 日志服务主体使用密钥的权限。此服务主体必须位于存储 KMS 密钥的同一 AWS 区域。

作为最佳实践，我们建议您将 KMS 密钥的使用限制在您指定的 AWS 账户或日志组中。

首先，`policy.json`使用以下get-key-policy命令保存 KMS 密钥的默认策略：

```
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`章节来增强 AWS KMS 密钥的安全性。有关更多信息，请参阅 AWS 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。

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

```
                "kms:Describe*"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:region:account-
id:log-group:log-group-name"
                }
            }
        }
    ]
}
```

本示例中的 Condition 部分对 AWS 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。

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

```
                "ArnLike": {
                    "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:region:account-
id:*"
                }
            }
        }
    ]
}
```

最后，使用以下put-key-policy命令添加更新的策略：

```
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。

要查看日志组是否已关联了 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

按以下方式使用 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

按以下方式使用 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

要取消与日志组关联的 KMS 密钥的关联，请使用以下disassociate-kms-key命令：

```
aws logs disassociate-kms-key --log-group-name my-log-group
```

AWS KMS 密钥和加密上下文

为了增强 AWS Key Management Service 密钥和加密日志组的安全性，Lo CloudWatch gs 现在将日志组 ARN 作为加密上下文的一部分，用于加密您的日志数据。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。加密上下文允许您使用 IAM 策略条件来限制 AWS 账户和日志组对 AWS 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：

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

```
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT-ID:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.region.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt*",
                "kms:Decrypt*",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:Describe*"
            ],
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "kms:EncryptionContext:aws:logs:arn":
 "arn:aws:logs:REGION:ACCOUNT-ID:log-
group:LOG-GROUP-NAME"
                }
            }
        }
    ]
}
```

5. 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 现在使用新密钥对所有新数据进行加密。

7. 接下来，从旧的密钥撤销除 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

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

9. 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

您可以使用 CloudWatch 日志组数据保护策略来帮助保护 Logs 摄取的敏感数据。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，与您选择的数据标识符匹配的敏感数据将在所有出口点被屏蔽，包括 Lo CloudWatch gs Insights、指标筛选器和订阅筛选器。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，并且还为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**
>
> 只有标准日志类中的日志组才支持对敏感数据进行屏蔽。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 日志支持许多托管数据标识符，这些标识符提供了预配置的数据类型，您可以选择这些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 日志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，检测还取决于在敏感数据附近找到某些关键字。您还可以使用自定义数据标识符来创建针对您的特定用例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 CloudWatch 时，就会发出一个指标。这是LogEventsWithFindings指标，它在 AWS/ Logs 命名空间中发布。您可以使用此指标来创建

CloudWatch 警报，也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 日志发送到的指标的更多信息 CloudWatch，请参阅使用 CloudWatch 指标进行监控。

每个托管数据标识符都旨在检测特定类型的敏感数据，例如信用卡号、 AWS 秘密访问密钥或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 日志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：

- 证书，例如私钥或私有访问 AWS 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
  - AWS CLI
    - AWS 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
- AWS CLI
  - AWS 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
  - 将审计结果发送到受保护的存储桶所需的密钥策略 AWS KMS
- 您可以保护的数据类型
  - CloudWatch 记录敏感数据类型的托管数据标识符
    - 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
    - 个人身份信息（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
      - 个人身份信息 (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 日志使用数据保护策略来选择要扫描的敏感数据，以及为保护这些数据而要采取的操作。要选择感兴趣的敏感数据，请使用数据标识符。 CloudWatch 记录数据保护，然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 Lo CloudWatch gs 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 日志应扫描的敏感数据。例如，姓名、地址或电话号码。
- 操作-后续操作，可以是 “审计” 或 “取消标识”。 CloudWatch 日志在发现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 L CloudWatch ogs 发布到 AWS/Logs 命名空间的LogEventsWithFindings指标。 CloudWatch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 日志发送到的指标的更多信息 CloudWatch，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`	*
创建以 CloudWatch 日志为审计目标的数据保护策略	`logs:PutAccountPolicy`	*
	`logs:PutDataProtectionPolicy`	*
	`logs:CreateLogDelivery`	*
	`logs:PutResourcePolicy`	*
	`logs:DescribeResourcePolicies`	*
	`logs:DescribeLogGroups`	*

操作	所需的 IAM 权限	资源
创建以 Firehose 为审计目标的数据保护政策	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*
	logs:CreateLogDelivery	*
	firehose:TagDeliveryStream	arn:aws:logs:::deliverystream/ *YOUR_DELIVERY_STREAM*
将 Amazon S3 作为审计目标来创建数据保护策略	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*
	logs:CreateLogDelivery	*
	s3:GetBucketPolicy	arn:aws:s3::: *YOUR_BUCKET*
	s3:PutBucketPolicy	arn:aws:s3::: *YOUR_BUCKET*
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	*

操作	所需的 IAM 权限	资源
	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: *YOUR_LOG_GROUP* :*
创建以 CloudWatch 日志为审计目标的数据保护策略	logs:PutDataProtectionPolicy logs:CreateLogDelivery logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :* * * * *

操作	所需的 IAM 权限	资源
创建以 Firehose 为审计目标的数据保护政策	logs:PutDataProtectionPolicy logs:CreateLogDelivery firehose:TagDeliveryStream	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :* * arn:aws:logs:::deliverystream/ *YOUR_DELIVERY_STREAM*
将 Amazon S3 作为审计目标来创建数据保护策略	logs:PutDataProtectionPolicy logs:CreateLogDelivery s3:GetBucketPolicy s3:PutBucketPolicy	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :* * arn:aws:s3::: *YOUR_BUCKET* arn:aws:s3::: *YOUR_BUCKET*
对已屏蔽的日志事件取消屏蔽	logs:Unmask	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :*
查看现有数据保护策略	logs:GetDataProtectionPolicy	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :*
删除数据保护策略	logs:DeleteDataProtectionPolicy	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :*

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。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "YOUR_SID_1",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:PutResourcePolicy",
                "logs:DescribeLogGroups",
                "logs:DescribeResourcePolicies"
            ],
            "Resource": "*"
        },
        {
            "Sid": "YOUR_SID_2",
            "Effect": "Allow",
            "Action": [
                "logs:GetDataProtectionPolicy",
                "logs:DeleteDataProtectionPolicy",
                "logs:PutDataProtectionPolicy",
                "s3:PutBucketPolicy",
                "firehose:TagDeliveryStream",
                "s3:GetBucketPolicy"
            ],
            "Resource": [
                "arn:aws:firehose:::deliverystream/YOUR_DELIVERY_STREAM",
                "arn:aws:s3:::YOUR_BUCKET",
                "arn:aws:logs:::log-group:YOUR_LOG_GROUP:*"
            ]
        }
    ]
}
```

创建账户范围的数据保护策略

您可以使用 CloudWatch 日志控制台或 AWS CLI 命令创建数据保护策略，以屏蔽您账户中所有日志组的敏感数据。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
- AWS CLI

控制台

使用控制台创建账户范围数据保护策略

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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. 在托管数据标识符中，为所有日志组选择要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）如果要使用自定义数据标识符来审计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正则表达式，用于在日志事件中搜索该类型的数据。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以包含 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必须不超过 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）。

AWS CLI

使用 AWS CLI 来创建数据保护策略

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
```

AWS CLI 或 API 操作的数据保护策略语法

当您创建用于 AWS 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 DataIdentifer 数组和带有 Audit 操作的 Operation 属性。DataIdentifer 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。如果您指定目标，例如日志组、Amazon Data Firehose 流和 S3 存储桶，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 DataIdentifer 数组和带有 Deidentify 操作的 Operation 属性。DataIdentifer 数组必须与策略第一个块中的 DataIdentifer 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。

以下是仅使用托管数据标识符的数据保护策略示例。此政策掩盖了电子邮件地址和美国驾照。

有关指定自定义数据标识符的策略的信息，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 日志控制台或 AWS CLI 命令创建数据保护策略来屏蔽敏感数据。

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
- AWS CLI

控制台

使用控制台创建数据保护策略

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，依次选择日志和日志组。
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. 在托管数据标识符中，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。

   有关您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）如果要使用自定义数据标识符来审计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正则表达式，用于在日志事件中搜索该类型的数据。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以包含 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必须不超过 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）。

AWS CLI

使用 AWS CLI 来创建数据保护策略

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
```

AWS CLI 或 API 操作的数据保护策略语法

当您创建用于 AWS 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 DataIdentifer 数组和带有 Audit 操作的 Operation 属性。DataIdentifer 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。如果您指定目标，例如日志组、Amazon Data Firehose 流和 S3 存储桶，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 DataIdentifer 数组和带有 Deidentify 操作的 Operation 属性。DataIdentifer 数组必须与策略第一个块中的 DataIdentifer 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 L CloudWatch ogs 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 日志见解查询语法。

- 将 GetLogEvents或 FilterLogEvents运算与unmask参数一起使用。

该CloudWatchLogsFullAccess策略包括logs:Unmask权限。要向没有 IAM 策略的用户授权 logs:Unmask CloudWatchLogsFullAccess，您可以向该用户附加自定义 IAM 策略。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

如果您将 CloudWatch 日志数据保护审计策略设置为将审计报告写入 CloudWatch 日志、Amazon S3 或 Firehose，则这些发现报告与以下示例类似。 CloudWatch 日志会为每个包含敏感数据的日志事件写入一份发现报告。

```
{
    "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。

将审计结果发送到受保护的存储桶所需的密钥策略 AWS KMS

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 一起使用的所需密钥策略的更多信息，请参阅 Amazon L CloudWatch ogs 用户指南Amazon S3中的。

您可以保护的数据类型

本节包含有关您可以在 CloudWatch 日志数据保护策略中保护的数据类型的信息。 CloudWatch 日志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您的特定用例量身定制的数据标识符。

目录

- CloudWatch 记录敏感数据类型的托管数据标识符
  - 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
- 个人身份信息（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
    - 个人身份信息 (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

CloudWatch 记录敏感数据类型的托管数据标识符

本节包含有关您可以使用托管数据标识符保护的数据类型以及哪些国家和地区与每种数据类型相关的信息。

对于某些类型的敏感数据， CloudWatch Logs Data Protection 会扫描数据附近的关键字，并仅在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 日志数据保护会自动匹配缺少空格或包含下划线 (_) 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解决关键字的常见变体。

下表列出了 CloudWatch 日志可以使用托管数据标识符检测到的凭证、设备、财务、医疗和受保护健康信息 (PHI) 的类型。除了特定类型的数据之外，这些数据可能也符合个人身份信息 (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
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
CepCode	个人	BR

标识符	类别	国家/地区和语言
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、 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

标识符	类别	国家/地区和语言
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
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 日志数据保护可以找到以下类型的凭据。

数据类型	数据标识符 ID	所需关键字	国家和地区
AWS 秘密访问密钥	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
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 日志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 日志数据保护可以找到以下类型的财务信息。

如果您设置了数据保护策略，则无论 CloudWatch 日志组位于哪个地理位置，日志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	注意
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	包括由最多 34 个字母数字字符组成的国际银行账

数据类型	数据标识符 ID	所需关键字	国家和地区	注意
				号（IBAN, International Bank Account Numbers），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	

数据类型	数据标识符 ID	所需关键字	国家和地区	注意
信用卡号	CreditCardNumber	account number, american express, 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	全部	检测要求数据为符合卢恩支票公式的13-19位数序列，并对以下任何类型的信用卡使用标准卡号前缀：美国运通、Dankort、Diner's Club、Discover、Electron、日本信用卡局（JCB）、万事达卡和Visa。UnionPay

数据类型	数据标识符 ID	所需关键字	国家和地区	注意
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

使用以下关键字检测由最多 34 个字母数字字符组成的国际银行账号（IBAN），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, kartennummer , kontonummer , kreditkartennummer , 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 日志不会报告以下序列的出现情况，信用卡发卡机构已保留这些顺序供公开测试。

```
122000000000003, 2222405343248877, 2222990905257051, 2223007648726984,
 2223577120017656,
30569309025904, 34343434343434, 3528000700000000, 3530111333300000, 3566002020360505,
 36148900647913,
36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237,
 4012888888881881,
4111111111111111, 4222222222222, 4444333322221111, 4462030000000000, 4484070000000000,
 4911830000000,
4917300800000000, 4917610000000000, 4917610000000000003, 5019717010103742,
 5105105105105100,
5111010030175156, 5185540810000019, 5200828282828210, 5204230080000017,
 5204740009900014, 5420923878724339,
5454545454545454, 5455330760000018, 5506900490000436, 5506900490000444,
 5506900510000234, 5506920809243667,
5506922400634930, 5506927427317625, 5553042241984105, 5555553753048194,
 5555555555554444, 5610591081018250,
6011000990139424, 6011000400000000, 6011111111111117, 630490017740292441,
 630495060000000000,
6331101999990016, 6759649826438453, 6799990100000000019, and 76009244561.
```

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 日志数据保护可以找到以下类型的受保护健康信息 (PHI)。

如果您设置了数据保护策略，则无论 CloudWatch 日志组位于哪个地理位置，日志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’assurance maladie , carte européenne d'assurance maladie , ceam, ehic, ehic#, finlandehicnumber# , gesundheitskarte , hälsokort , health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte , krankenversicherungsnummer , medical account number, numero conto medico, numéro d’assurance maladie , numéro de carte d’assurance , numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin , sairausvakuutuskortti , sairausvakuutusnumero , 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, terveyskortti , tessera sanitaria assicurazione numero , versicherungsnummer	
健康保险索赔编号 (HICN, Health Insurance Claim Number)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#, hicno#	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, Healthcare Common Procedure Coding System) 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 beneficiary	美国
国家药品编码 (NDC, National Drug Code)	NationalDrugCode	national drug code, ndc	美国
国家提供商识别码 (NPI, National Provider Identifier)	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 日志数据保护可以找到以下类型的个人身份信息 (PII)。

如果您设置了数据保护策略，则无论 CloudWatch 日志组位于哪个地理位置，日志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	注意
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	注意
				杠（/）或连字符（-）分隔。
Código de Endereçamento Postal (CEP)	CepCode	cep, código de endereçamento postal, codigo 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 fisicas, 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	注意
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	ElectoralRollNumber	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	注意
国家统计与经济研究所 (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字）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字）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, National Insurance Number)	`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	注意
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	注意
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	注意
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	注意
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	注意
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	注意
全球定位系统 (GPS, Global Positioning System) 坐标	LatLong	coordinate ，coordinates ，lat long， latitude longitude ，location，position	任何	CloudWatch 如果纬度和经度坐标成对存储并且采用十进制 (DD) 格式，例如 41.948614、-87.655311，则日志可以检测 GPS 坐标。不支持采用度十进制分 (DDM, Degrees Decimal Minutes) 格式的坐标，例如 41°56.9168'N 87°39.3187'W，

数据类型	数据标识符 ID	所需关键字	国家和地区	注意
				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 日志只能检测全名。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	注意
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 日志可以检测由 17 个字符的序列组成并符合 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码，CloudWatch Logs 要求关键字与数字相近。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 osterreich
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 prúkaz, 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	δεια οδήγησης, adeia odigisis
匈牙利	illesztőprogramok lic, jogosítvány, jogsi, 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 conducer, el carnet de conducer, licencia conducer, licencia de manejo, número carnet conducer, número de carnet de conducer, número de permiso conducer, número de permiso de conducer, número licencia conducer, número permiso conducer, permiso conducción, permiso conducer, permiso de conducción
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字

为了检测各种类型的国民身份证号码， CloudWatch Logs 要求关键字与这些数字非常接近。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’identité, 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 fiscal, 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

要检测各种类型的护照号码，CloudWatch Logs 要求关键字与护照号码相近。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	passeport, passeport#, 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, reisepass, reisepass–nr, reisepassnummer
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身份和参考号， CloudWatch Logs 要求关键字与数字相近。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 fisica, cadastro de pessoas físicas, cadastro de pessoas fisicas, cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, 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, steueride ntifikationsnummer, 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身份信息（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 日志自定义数据标识符有以下限制：

- 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别的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别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

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即可。例如，您可以输入**Employee_ID**名称和**EmployeeID-\d{9}**作为正则表达式。此正则表达式将检测并屏蔽后面有九个数字的日志事件EmployeeID-。例如，EmployeeID-123456789

在数据保护策略中使用自定义数据标识符

如果您使用 AWS CLI 或 AWS API 来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和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后面EmployeeID-有九位数的日志事件EmployeeID-。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）根据需要继续向 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。

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器来搜索和筛选进入 CloudWatch 日志的日志数据。指标筛选器定义了发送到日志时要 CloudWatch 在日志数据中查找的术语和模式。 CloudWatch 日志使用这些指标筛选器将日志数据转换为数字 CloudWatch 指标，您可以绘制图表或设置警报。

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。

> **Note**
>
> 仅标准日志类中的日志组支持指标筛选器。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，包括百分位数统计信息。

> **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。筛选结果返回前 50 行，如果筛选结果上的时间戳早于指标创建时间，则不会显示这些行。

内容

- 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

对 CloudWatch 日志应如何解释每个日志事件中的数据的符号描述。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

应向其发布监控日志信息的 CloudWatch 指标的名称。例如，您可以发布到名为的指标 ErrorCount。

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**
>
> 指标筛选器有何不同 CloudWatch 日志 Insights 查询

指标筛选器与 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器中。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 Logs Insights 查询语言查询 CloudWatch 日志组的信息，请参阅CloudWatch 日志见解查询语法。
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

指标筛选器允许您搜索和筛选进入日志的 CloudWatch 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 日志量度。当日志数据发送到日志时，您可以定义要 CloudWatch 在日志数据中查找的术语和模式。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 在您的录入事件中出现的次数，则可以指定名为 `ErrorCode` 的维度，以显示包含单词 ERROR 的录入事件总数，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 支持的商品的完整列表，请参阅 Amazon CloudWatch API 参考MetricDatum中的。

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。如果您未指定默认值，则当您的指标筛选条件未找到匹配项时，将 CloudWatch 不会报告数据。即使该值为 0，也建议您指定原定设置值。设置默认值有助于更准确地 CloudWatch 报告数据，并 CloudWatch 防止聚合参差不齐的指标。 CloudWatch 每分钟汇总和报告指标值。

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。如果您的指标筛选条件未找到匹配项，则会 CloudWatch 报告该指标的默认值。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 AWS CLI 创建指标筛选条件，以发布包含 JSON 和以空格分隔的日志事件生成的指标的维度。维度是名称/值值对，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@@ 尺寸
- 示例：从 Apache 日志中提取字段并在 A mazon CloudWatch 日志用户指南中分配维度

> **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，请参阅创建账单警报以监控您的预估 AWS 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可以如何将 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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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。

   有关如何创建日志组的信息，请参阅 Amazon 日志用户指南中的 CloudWatch 日志中的创建 CloudWatch 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。设置默认值有助于更准确地 CloudWatch 报告数据，并 CloudWatch 防止聚合参差不齐的指标。 CloudWatch 每分钟汇总和报告指标值。
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器

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。

2. 在导航窗格中，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）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。
7. 选择 Next（下一步），然后为指标筛选条件输入名称。
8. 在指标详细信息下，在指标命名空间中，输入要发布指标的 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）。
9. 对于 Metric name（指标名称），为新指标输入名称。
10. 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。
11. （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。
12. （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。

13. 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 MyAppAccessCount 为的指标筛选器应用于日志组 MyApp /access.log，以便在 CloudWatch命名空间 EventCount 中创建指标 MyNamespace。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器

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择 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）。

要使用创建指标筛选器 AWS CLI

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。您应该会看到发布到该指标的数据点 MyAppAccessEventCount。

要使用发布事件数据 AWS CLI

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。 CloudWatch 对于每个包含 Error 的事件，Logs 会 ErrorCount 在 MyApp/message.log 命名空间中向 CloudWatch 自定义指标发布一个数据点，其值为 “1”。如果没有任何事件包含“Error”这个单词，则发布值 0。在 CloudWatch控制台中绘制这些数据时，请务必使用总和统计数据。

创建指标筛选条件后，可以在 CloudWatch 控制台中查看该指标。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器

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择 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 “指标详细信息” 下的 “指标命名空间” 中，键入MyNameSpace。
9. 对于 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）。

要使用创建指标筛选器 AWS CLI

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。

要使用发布活动 AWS CLI

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 日志，您可以监控 Apache 服务器返回 HTTP 404 响应的次数，这是未找到页面的响应代码。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和协议
- 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器

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
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 Nam（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 **`ApacheNotFoundErrorCount`**。
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）。

要使用创建指标筛选器 AWS CLI

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=ApacheNotFoundErrorCount,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": "ApacheNotFoundErrorCount"
                }
            ],
            "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"
```

放置这些示例日志事件后不久，您就可以将 CloudWatch 控制台中命名的指标检索为 ApacheNotFoundErrorCount。

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器

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择 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）。

要使用创建指标筛选器 AWS CLI

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器

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择 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）。

要创建此指标筛选器，请使用 AWS CLI

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,dimensio
$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 通话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器

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择 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。

要列出指标筛选条件，请使用 AWS CLI

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": "ApacheNotFoundErrorCount"
                }
            ],
            "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器

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择 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）。

要删除指标筛选条件，请使用 AWS CLI

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

您可以使用订阅来访问来自 CloudWatch 日志的实时日志事件源，并将其传输到其他服务，例如 Amazon Kinesis 流、Amazon Data Firehose 流，或者 AWS Lambda 用于自定义处理、分析或加载到其他系统。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。

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。订阅筛选器定义了筛选器模式，用于筛选哪些日志事件会传递到您的 AWS 资源，以及有关将匹配的日志事件发送到何处的信息。

您可以在账户级别和日志组级别创建订阅。每个账户可以有一个账户级别的订阅过滤器。每个日志组最多只能有两个与其关联的订阅筛选条件。

> **Note**
>
> 如果目标服务返回可重试的错误，例如限制异常或可重试的服务异常（例如 HTTP 5xx），则 CloudWatch 日志会继续重试传送长达 24 小时。 CloudWatch 如果错误是不可重试的错误（例如或），则日志不会尝试重新传送。 AccessDeniedException ResourceNotFoundException在这些情况下，订阅筛选器最多会被禁用 10 分钟，然后 CloudWatch Logs 会重试向目标发送日志。在此禁用期内，将跳过日志。

CloudWatch 日志还会生成有关将日志事件转发给订阅的 CloudWatch 指标。有关更多信息，请参阅 使用 CloudWatch 指标进行监控。

您还可以使用 CloudWatch 日志订阅将日志数据近乎实时地流式传输到 Amazon S OpenSearch ervice 集群。有关更多信息，请参阅将 CloudWatch 日志数据流式传输到 Amazon OpenSearch 服务。

只有标准日志类中的日志组才支持订阅。有关日志类的更多信息，请参阅日志类。

> **Note**
>
> 订阅过滤器可能会批量记录事件，以优化传输并减少拨打到目标的呼叫量。不能保证批处理，但会尽可能使用批处理。

内容

- 概念
- 记录组级订阅过滤器
- 账户级订阅过滤器
- 跨账户跨区域订阅
- 混淆代理问题防范
- 日志递归防护

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

对 CloudWatch 日志应如何解释每个日志事件中的数据的符号描述，以及限制传送到目标 AWS 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

您要用作订阅源目标的 Kinesis Data Streams 流、Firehose 流或 Lambda 函数的亚马逊资源名称 (ARN)。

角色 ARN

一个 IAM 角色，它向 CloudWatch Logs 授予将数据放入所选目标所需的权限。Lambda 目标不需要此角色，因为 CloudWatch 日志可以从 Lambda 函数本身的访问控制设置中获得必要的权限。

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，还包括以下关键元素：

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。

对于账户级订阅，还包括以下关键元素：

选择标准

用于选择哪些日志组应用了账户级订阅筛选器的标准。如果您未指定此项，则账户级别的订阅筛选器将应用于该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，请参阅日志递归防护。

选择标准的大小限制为 25 KB。

记录组级订阅过滤器

你可以将订阅过滤器与 Kinesis Data Streams、Lambda 或 Firehose 一起使用。通过订阅筛选条件发送到接收服务的日志将经过 base64 编码并使用 gzip 格式进行压缩。

您可以使用筛选条件和模式语法搜索日志数据。

示例

- 示例 1：Kinesis Data Streams 订阅筛选条件
- 示例 2：带有以下内容的订阅过滤器 AWS Lambda
- 示例 3：使用 Amazon Data Firehose 的订阅筛选条件

示例 1：Kinesis Data Streams 订阅筛选条件

以下示例将订阅过滤器与包含 AWS CloudTrail 事件的日志组相关联。订阅过滤器会将通过 “根” AWS 凭据记录的所有活动发送到 Kinesis Data Streams 中名为 “RootAccess” 的直播中。有关如何向 CloudWatch 日志发送 AWS CloudTrail 事件的更多信息，请参阅AWS CloudTrail 用户指南中的向 CloudWatch 日志发送 CloudTrail 事件。

> **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或指定`randomdistribution`何时创建订阅筛选器 put-subscription-filter。默认情况下，流过滤器按日志流分布，这可能会导致限制。

- 使用 CloudWatch 指标监控您的直播。这可以帮助您识别任何节流并相应地调整配置。例如，该`DeliveryThrottling`指标可用于跟踪在将数据转发到订阅 CloudWatch 目标时限制日志的日志事件数量。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 中的直播容量相匹配。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）。你可以使用以下 Kinesis Data St reams describe-stream 命令来检查。StreamDescription StreamStatus财产。此外，请注意 StreamDescription.streamArn 的值，因为您将在以后的步骤中使用它：

```
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，该角色将授予 CloudWatch 日志将数据放入您的直播的权限。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 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": "AAOIIAH450GAB4HC5F431",
        "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 CloudWatch 日志可以对您的账户执行的操作。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. 在直播处于 “活动” 状态且您已创建 IAM 角色后，您可以创建 CloudWatch 日志订阅筛选器。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. 设置订阅过滤器后，Lo CloudWatch gs 会将所有与过滤器模式匹配的传入日志事件转发到您的直播中。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/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+OIK8zM5My8ID
+g6rMo7UKWeI4+IWiK2OSh0uP"
}
```

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

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

原始日志数据的 AWS 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， CloudWatch 日志可能会发出“CONTROL_MESSAGE” 类型的 Kinesis Data Streams 记录，主要用于检查目标是否可达。

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：带有以下内容的订阅过滤器 AWS Lambda

在此示例中，您将创建一个 Lo CloudWatch gs 订阅过滤器，用于将日志数据发送到您的 AWS 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 AWS Lambda 限制。

为 Lambda 创建订阅筛选条件

1. 创建 AWS Lambda 函数。

   确保您已设置 Lambda 执行角色。有关更多信息，请参阅 AWS 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 CloudWatch 予 Logs 执行函数的权限。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\":
\"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

原始日志数据的 AWS 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， CloudWatch 日志可能会发出“CONTROL_MESSAGE” 类型的 Lambda 记录，主要用于检查目标是否可达。

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：使用 Amazon Data Firehose 的订阅筛选条件

在此示例中，您将创建一个 CloudWatch 日志订阅，该订阅将与您定义的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。从 CloudWatch 日志发送到 Amazon Data Firehose 的数据已使用 gzip 第 6 级压缩进行压缩，因此您无需在 Firehose 传输流中使用压缩。然后，您可以使用 Firehose 中的解压缩功能自动解压缩日志。有关更多信息，请参阅使用日志写入 Kinesis Data CloudWatch Fire hose。

> **Note**
>
> 在创建 Firehose 流之前，请计算将生成的日志数据量。请务必创建能够处理此音量的 Firehose 直播。如果流无法处理卷，则日志流将受限制。有关 Firehose 直播容量限制的更多信息，请参阅亚马逊数据 Firehose 数据限制。

为 Firehose 创建订阅过滤器

1. 创建 Amazon Simple Storage Service (Amazon S3) 存储桶 我们建议您使用专为 Logs 创建的存储 CloudWatch 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 my-bucket --create-bucket-configuration
 LocationConstraint=region
```

下面是示例输出：

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

2. 创建 IAM 角色，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，请参阅亚马逊数据 Fireh ose 开发者指南中的使用亚马逊数据 Fireh ose 控制访问权限。

   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": "AAOIIAH450GAB4HC5F431",
        "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 可以对您的账户执行哪些操作。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:::my-bucket",
          "arn:aws:s3:::my-bucket/*" ]
    }
  ]
}
```

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. 按如下方式创建目标 Firehose 交付流，将 roLearn 和 Bucketarn 的占位符值替换为您创建的角色和存储桶 AR N：

```
aws firehose create-delivery-stream \
   --delivery-stream-name 'my-delivery-stream' \
   --s3-destination-configuration \
```

```
  '{"RoleARN": "arn:aws:iam::123456789012:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::my-bucket"}'
```

请注意，对于已交付的亚马逊 S3 对象，Firehose 会自动使用 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)。你可以使用 Fire describe-delivery-streamhose 命令来检查。DeliveryStreamDescription DeliveryStreamStatus财产。此外，请注意DeliveryStreamDescription。 DeliveryStreamARN 值，因为你将在后面的步骤中需要它：

```
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-000000000001",
                "S3DestinationDescription": {
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "RoleARN": "delivery-stream-role",
                    "BucketARN": "arn:aws:s3:::my-bucket",
                    "BufferingHints": {
                        "IntervalInSeconds": 300,
                        "SizeInMBs": 5
                    }
                }
            }
        ]
    }
}
```

8. 创建 IAM 角色，该角色向 CloudWatch 日志授予将数据放入 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 CloudWatch 日志可以对您的账户执行的操作。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
```

12. 在 Amazon Data Firehose 传输流处于活动状态并且您已创建 IAM 角色之后，您可以创建 CloudWatch 日志订阅筛选条件。订阅筛选器会立即启动实时日志数据从所选日志组流向您的 Amazon Data Firehose 传输流：

```
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. 设置订阅筛选器后，Lo CloudWatch gs 会将所有与筛选模式匹配的传入日志事件转发到您的 Amazon Data Firehose 传输流。您的数据将根据在 Amazon Data Firehose 传输流中设置的时间

缓冲间隔开始出现在您的 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 'my-bucket' --prefix 'firehose/'
{
    "Contents": [
        {
            "LastModified": "2015-10-29T00:01:25.000Z",
            "ETag": "\"a14589f8897f4089d3264d9e2d1f1610\"",
            "StorageClass": "STANDARD",
            "Key": "firehose/2015/10/29/00/my-delivery-stream-2015-10-29-00-01-21-
a188030a-62d2-49e6-b7c2-b11f1a7ba250",
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b5"
            },
            "Size": 593
        },
        {
            "LastModified": "2015-10-29T00:35:41.000Z",
            "ETag": "\"a7035b65872bb2161388ffb63dd1aec5\"",
            "StorageClass": "STANDARD",
            "Key": "firehose/2015/10/29/00/my-delivery-
stream-2015-10-29-00-35-40-7cc92023-7e66-49bc-9fd4-fc9819cc8ed3",
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b6"
            },
            "Size": 5752
        }
    ]
}
```

```
aws s3api get-object --bucket 'my-bucket' --key 'firehose/2015/10/29/00/my-
delivery-stream-2015-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250'
 testfile.gz

{
    "AcceptRanges": "bytes",
    "ContentType": "application/octet-stream",
    "LastModified": "Thu, 29 Oct 2015 00:07:06 GMT",
    "ContentLength": 593,
```

```
    "Metadata": {}
}
```

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
```

账户级订阅过滤器

Important

订阅过滤器存在导致无限递归循环的风险，如果不加以解决，可能会导致摄取账单的大幅增加。为了降低这种风险，我们建议您在账户级订阅筛选器中使用选择标准，将从订阅交付工作流程中的资源中提取日志数据的日志组排除在外。有关此问题以及确定要排除哪些日志组的更多信息，请参阅日志递归防护。

您可以设置账户级别的订阅策略，其中包括账户中日志组的子集。账户订阅政策可以与 Kinesis Data Streams、Lambda 或 Firehose 配合使用。通过账户级订阅策略发送到接收服务的日志采用 gzip 格式进行 base64 编码和压缩。

Note

要查看您账户中所有订阅筛选策略的列表，请使用`--policy-type`参数值`SUBSCRIPTION_FILTER_POLICY`为的`describe-account-policies`命令。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：带有以下内容的订阅过滤器 AWS Lambda
- 示例 3：使用 Amazon Data Firehose 的订阅筛选条件

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，请先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。如果直播没有足够的分片，则会对其进行限制。有关直播量限制的更多信息，请参阅 Kinesis Data Streams 文档中的配额和限制。

> **Warning**
>
> 由于多个日志组的日志事件会转发到目标，因此存在限制的风险。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：
>
> - 使用指标监控你的 Kinesis Data Streams CloudWatch 直播。这可以帮助您识别限制并相应地调整配置。例如，该`DeliveryThrottling`指标跟踪在将数据转发到订阅目标时限制 CloudWatch 日志的日志事件的数量。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，请参阅按需模式。
> - 限制您的 CloudWatch 日志订阅筛选模式，使其与 Kinesis Data Streams 中的直播容量相匹配。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 中的数据流。过滤器模式将任何日志事件与文本进行匹配，`Test`并将其转发到 Kinesis Data Streams 中的数据流。

为 Kinesis Data Streams 创建账户级订阅政策

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，直至直播变为活跃状态。您可以使用 desc ribe-stream 命令检查直播是否处于活动状态。StreamDescription StreamStatus财产。

```
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": "EXAMPLE84634633746074317682114555",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
                    "EXAMPLE688818456679503831981458784591352702181572610"
                }
            }
        ]
    }
}
```

3. 创建 IAM 角色，该角色将授予 CloudWatch 日志将数据放入您的直播的权限。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 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 CloudWatch 日志可以对您的账户执行的操作。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 控制台进行创建。

```
{
  "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. 在直播处于 “活动” 状态且您已创建 IAM 角色后，您可以创建 CloudWatch 日志订阅筛选器策略。该策略会立即启动实时日志数据流向您的数据流。在此示例中，所有包含该字符串的日志事件ERROR都将流式传输，但名为LogGroupToExclude1和的日志组中的日志事件除外LogGroupToExclude2。

```
aws logs put-account-policy \
    --policy-name "ExamplePolicy" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document '{"RoleArn":"arn:aws:iam::123456789012:role/
CWLtoKinesisRole", "DestinationArn":"arn:aws:kinesis:region:123456789012:stream/
TestStream", "FilterPattern": "Test", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

8. 设置订阅过滤器后，Lo CloudWatch gs 会将所有符合过滤模式和选择标准的传入日志事件转发到您的直播中。

   该selection-criteria字段是可选的，但对于从订阅筛选器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，请参阅日志递归防护。目前，NOT IN 是唯一支持的运算符selection-criteria。

   您可以使用 Kinesis Data Streams 分片迭代器并使用 Kinesis Data Streams 命令获取一些 Kinesis Data Streams get-records 记录来验证日志事件的流：

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

```
{
    "ShardIterator":
    "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL
+wev+e2P4djJg4L9wmXKvQYoE+rMUiFq
```

```
+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f+OIK8zM5My8ID
+g6rMo7UKWeI4+IWiK2OSh0uP"
}
```

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

在 Kinesis Data Streams 开始返回数据之前，你可能需要多次使用此命令。

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：

```
{
    "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"
}
```

数据结构中的关键元素如下：

messageType

数据消息将使用“DATA_MESSAGE”类型。有时， CloudWatch 日志可能会发出“CONTROL_MESSAGE” 类型的 Kinesis Data Streams 记录，主要用于检查目标是否可达。

owner

原始日志数据的 AWS 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。

策略级别

政策的执行级别。“账户级别_POLICY” 是`policyLevel`针对账户级别的订阅筛选器策略。

示例 2：带有以下内容的订阅过滤器 AWS Lambda

在此示例中，您将创建一个 CloudWatch 日志账户级别的订阅筛选器策略，该策略将日志数据发送到您的 AWS 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音量，则日志流将被限制。由于所有日志组或账户日志组子集的日志事件都会转发到目标，因此存在限制的风险。有关 Lambda 限制的更多信息，请参阅 AWS Lambda 限制。

为 Lambda 创建账户级订阅筛选策略

1. 创建 AWS Lambda 函数。

   确保您已设置 Lambda 执行角色。有关更多信息，请参阅 AWS 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 CloudWatch 予 Logs 执行函数的权限。使用以下命令，将占位符帐户替换为您自己的帐户。

```
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自己的账户。在此示例中，所有包含该字符串的日志事件ERROR都将流式传输，但名为LogGroupToExclude1和的日志组中的日志事件除外LogGroupToExclude2。

```
aws logs put-account-policy \
    --policy-name "ExamplePolicyLambda" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document
 '{"DestinationArn":"arn:aws:lambda:region:123456789012:function:helloWorld",
 "FilterPattern": "Test", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

设置订阅过滤器后，Lo CloudWatch gs 会将所有符合过滤模式和选择标准的传入日志事件转发到您的直播中。

该selection-criteria字段是可选的，但对于从订阅筛选器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，请参阅日志递归防护。目前，NOT IN 是唯一支持的运算符selection-criteria。

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不会应用于目标 Lambda 函数的日志组。这是为了防止可能导致摄取账单增加的无限日志递归。有关此问题的更多信息，请参阅日志递归防护。

数据结构中的关键元素如下：

messageType

数据消息将使用“DATA_MESSAGE”类型。有时， CloudWatch 日志可能会发出“CONTROL_MESSAGE” 类型的 Kinesis Data Streams 记录，主要用于检查目标是否可达。

owner

原始日志数据的 AWS 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。

策略级别

政策的执行级别。“账户级别_POLICY” 是`policyLevel`针对账户级别的订阅筛选器策略。

示例 3：使用 Amazon Data Firehose 的订阅筛选条件

在此示例中，您将创建 CloudWatch 日志账户级别的订阅筛选器策略，该策略将与您定义的筛选条件匹配的传入日志事件发送到您的 Amazon Data Firehose 传输流。从 CloudWatch 日志发送到 Amazon Data Firehose 的数据已使用 gzip 第 6 级压缩进行压缩，因此您无需在 Firehose 传输流中使用压缩。

然后，您可以使用 Firehose 中的解压缩功能自动解压缩日志。有关更多信息，请参阅使用日志写入 Kinesis Data CloudWatch Fire hose。

> **Warning**
>
> 在创建 Firehose 流之前，请计算将生成的日志数据量。请务必创建能够处理此音量的 Firehose 直播。如果流无法处理卷，则日志流将受限制。有关 Firehose 直播容量限制的更多信息，请参阅亚马逊数据 Firehose 数据限制。

为 Firehose 创建订阅过滤器

1. 创建 Amazon Simple Storage Service (Amazon S3) 存储桶 我们建议您使用专为 Logs 创建的存储 CloudWatch 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 my-bucket --create-bucket-configuration
    LocationConstraint=region
   ```

   下面是示例输出：

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

2. 创建 IAM 角色，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，请参阅亚马逊数据 Fireh ose 开发者指南中的使用亚马逊数据 Fireh ose 控制访问权限。

   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，因为你将在后面的步骤中需要它：

```
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 可以对您的账户执行哪些操作。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:::my-bucket",
            "arn:aws:s3:::my-bucket/*" ]
    }
  ]
}
```

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. 按如下方式创建目标 Firehose 交付流，将 roLearn 和 Bucketarn 的占位符值替换为您创建的角色和存储桶 AR N：

```
aws firehose create-delivery-stream \
   --delivery-stream-name 'my-delivery-stream' \
   --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::123456789012:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::my-bucket"}'
```

nFireHose 会自动对已交付的 Amazon S3 对象使用 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，直至直播变为活跃状态。你可以使用 Fire describe-delivery-streamhose 命令来检查。DeliveryStreamDescription DeliveryStreamStatus财产。此外，请注意DeliveryStreamDescription。 DeliveryStreamARN 值，因为你将在后面的步骤中需要它：

```
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-000000000001",
```

```
                "S3DestinationDescription": {
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "RoleARN": "delivery-stream-role",
                    "BucketARN": "arn:aws:s3:::my-bucket",
                    "BufferingHints": {
                        "IntervalInSeconds": 300,
                        "SizeInMBs": 5
                    }
                }
            }
        ]
    }
}
```

8. 创建 IAM 角色，该角色向 CloudWatch 日志授予将数据放入 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，因为你将在后面的步骤中使用它：

```
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 CloudWatch 日志可以对您的账户执行的操作。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
```

12. 在 Amazon Data Firehose 传输流处于活动状态并且您已创建 IAM 角色之后，您可以创建 CloudWatch 日志账户级别的订阅筛选器策略。该策略会立即启动实时日志数据从所选日志组流向您的 Amazon Data Firehose 传输流：

```
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. 设置订阅筛选器后，Lo CloudWatch gs 会将与筛选模式匹配的传入日志事件转发到您的 Amazon Data Firehose 传输流。

    该`selection-criteria`字段是可选的，但对于从订阅筛选器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，请参阅日志递归防护。目前，NOT IN 是唯一支持的运算符`selection-criteria`。

    您的数据将根据在 Amazon Data Firehose 传输流中设置的时间缓冲间隔开始出现在您的 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 'my-bucket' --prefix 'firehose/'
{
    "Contents": [
        {
            "LastModified": "2023-10-29T00:01:25.000Z",
            "ETag": "\"a14589f8897f4089d3264d9e2d1f1610\"",
            "StorageClass": "STANDARD",
            "Key": "firehose/2015/10/29/00/my-delivery-stream-2015-10-29-00-01-21-
a188030a-62d2-49e6-b7c2-b11f1a7ba250",
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b5"
            },
            "Size": 593
        },
        {
            "LastModified": "2015-10-29T00:35:41.000Z",
```

```
            "ETag": "\"a7035b65872bb2161388ffb63dd1aec5\"",
            "StorageClass": "STANDARD",
            "Key": "firehose/2023/10/29/00/my-delivery-stream-2023-10-29-00-35-40-
EXAMPLE-7e66-49bc-9fd4-fc9819cc8ed3",
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "EXAMPLE6be062b19584e0b7d84ecc19237f87b6"
            },
            "Size": 5752
        }
    ]
}
```

```
aws s3api get-object --bucket 'my-bucket' --key 'firehose/2023/10/29/00/my-
delivery-stream-2023-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250'
 testfile.gz

{
    "AcceptRanges": "bytes",
    "ContentType": "application/octet-stream",
    "LastModified": "Thu, 29 Oct 2023 00:07:06 GMT",
    "ContentLength": 593,
    "Metadata": {}
}
```

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

您可以与其他 AWS 账户的所有者协作，并在您的 AWS 资源上接收他们的日志事件，例如 Amazon Kinesis 或 Amazon Data Firehose 流（这称为跨账户数据共享）。例如，可以从集中式 Kinesis Data Streams 或 Firehose 流中读取这些日志事件数据，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 AWS 区域。但是，目的地指向的 AWS 资源可以位于不同的区域。在以下各节的示例中，所有特定于地区的资源都是在美国东部（弗吉尼亚北部）创建的。

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 的跨账户跨区域账户级别订阅
- 使用 Firehose 进行跨账户跨区域账户级别订阅

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：

- 日志数据发送者-从收件人那里获取目标信息，并让 Lo CloudWatch gs 知道它已准备好将其日志事件发送到指定目标。在本节其余部分的步骤中，显示日志数据发送者的虚构 AWS 账号为 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-设置封装 Kinesis Data Streams 流的目标，并 CloudWatch 让日志知道接收者想要接收日志数据。接收者随后与发送者共享与此目标有关的信息。在本节其余部分的步骤中，显示日志数据接收者的虚构 AWS 账号为 999999999999。

要开始接收来自跨账户用户的日志事件，日志数据接收者首先创建 CloudWatch 日志目标。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 Feed 目标的 AWS 资源的亚马逊资源名称 (ARN)。

角色 ARN

一个 AWS Identity and Access Management (IAM) 角色，它向 CloudWatch Logs 授予将数据放入所选数据流的必要权限。

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 AWS 区域。但是，目标指向的 AWS 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，日志数据接收者帐户的 AWS 帐户 ID 为 999999999999，而日志数据发送者 AWS 帐户 ID 为 111111111111。

此示例使用 RecipientStream名为的 Kinesis Data Streams 流创建一个目标，以及一个 CloudWatch 允许日志向其写入数据的角色。

创建目标后， CloudWatch Logs 会代表收件人账户向目标发送一条测试消息。当订阅筛选器稍后处于活动状态时，Logs 会代表源账户将 CloudWatch 日志事件发送到目标。

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。你可以使用 a ws kinesis describe-stream 命令来检查。StreamDescription StreamStatus财产。此外，请记下 StreamDescription.streamArn 的值，因为您稍后会将其传递给 Logs： CloudWatch

```
aws kinesis describe-stream --stream-name "RecipientStream"
{
  "StreamDescription": {
    "StreamStatus": "ACTIVE",
    "StreamName": "RecipientStream",
    "StreamARN": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "340282366920938463463374607431768EXAMPLE",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49551135218688818456679503831981458784591352702 18EXAMPLE"
```

```
        }
      }
    ]
  }
}
```

您的流可能需要一两分钟才会以活动状态显示。

3. 创建 IAM 角色以授予 CloudWatch Logs 将数据放入您的直播的权限。首先，你需要在文件 ~/TrustPolicyFor cwl .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。请记下返回的 Role.Arn 值，因为它稍后也会传递给 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": "AAOIIAH450GAB4HC5F431",
        "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 日志可以对您的账户执行哪些操作。首先，使用文本编辑器在文件 ~/ PermissionsFor cwl .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 a w s 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. 在直播处于活动状态并且您已创建 IAM 角色之后，您可以创建 CloudWatch 日志目标。

   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。记下有效载荷中返回的：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 action，并向发件人账户授予访问目标的权限。

   该策略向发送日志的 AWS 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。

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

```
            "Condition": {
               "StringEquals" : {
                   "aws:PrincipalOrgID" : ["o-1234567890"]
                }
            }
        }
    ]
}
```

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。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111111111111"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" :
 "arn:aws:logs:region: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
```

####### AWS ###### 111111111111 ##### ARN arn: aws: logs: region: 999999999999: Destinationation: testDestination: te ***PutSubscriptionFilter****stDestination ######*任何其他用户试图拨 PutSubscriptionFilter 打该目的地的电话都将被拒绝。

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，如果您使用 AWS Organizations 的是跨账户权限，请按照中的步骤 2：（仅在使用企业时）创建 IAM 角色步骤操作。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。

本节中的步骤创建一个 IAM 角色，该角色 CloudWatch 可以假设和验证发件人账户是否有权针对收件人目的地创建订阅筛选条件。

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 AWS 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 可以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 AWS 跨账户策略评估逻辑，要访问任何跨账户资源（例如用作订阅过滤器目标的 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 AWS 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*RoleName*或表示的值*UserName*。

2. 登录发送账户并使用您在步骤 1 AWS Management Console 中输入的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。下面的示例策略显示了建议的权限。

   以下策略仅允许在单个 AWS 账户（账户）中对任何目标资源创建订阅筛选器123456789012：

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

   以下策略仅允许在单个 AWS 账户（账户）sampleDestination中命名的特定目标资源上创建订阅筛选器123456789012：

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

```
                "arn:aws:logs:*:123456789012:destination:sampleDestination"
            ]
        }
    ]
}
```

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 AWS 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，订阅筛选器是在发送账户中创建的。该过滤器与包含 AWS CloudTrail 事件的日志组相关联，因此 “根” AWS 凭据记录的每个活动都将传送到您之前创建的目标。该目的地封装了一个名为 “” 的流。RecipientStream

以下各节的其余步骤假设您已按照AWS CloudTrail 用户指南中向 CloudWatch 日志发送 CloudTrail事件中的说明进行操作，并创建了一个包含您的 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 AWS 区域。但是，目标可以指向位于不同区域的 AWS 资源，例如 Kinesis Data Streams 流。

验证日志事件流

创建订阅过滤器后，Lo CloudWatch gs 会将所有与过滤器模式匹配的传入日志事件转发到封装在名为“” 的目标流中的流。RecipientStream目标所有者可以通过使用 aws kinesis get-shard-iterator 命令获取 Kinesis Data Streams 分片，然后使用 aw s kinesis get-records 命令获取一些 Kinesis Data Streams 记录来验证是否发生了这种情况：

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

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

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

> **Note**
>
> 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": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        }
    ]
}
```

此数据结构中的键元素如下所示：

owner

原始日志数据的 AWS 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， CloudWatch 日志可能会发出“CONTROL_MESSAGE” 类型的 Kinesis Data Streams 记录，主要用于检查目标是否可达。

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. 获取当前与目标 TestDest ination 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。将此政策放入 ~/ NewAccessPolicy .json 文件中：

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

```
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "222222222222"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:region:999999999999:destination:testDestination"
    }
  ]
}
```

3. 调用PutDestinationPolicy将 NewAccessPolicy.json 文件中定义的策略与目标关联起来：

```
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 启用来自 AWS 服务的日志记录 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。此更新添加了一个 IAM 角色，该角色 CloudWatch 可以假设并验证发件人账户有权向收件人账户发送日志。

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 可以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。

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

2. 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 AWS 账户 ID 的访问策略，请包括 force 参数。

> **Warning**
>
> 如果您正在处理中列出的 AWS 服务发送的日志启用来自 AWS 服务的日志记录，则在执行此步骤之前，必须先更新所有发件人账户中的订阅筛选器，如中所述步骤 1：更新订阅筛选条件。

```
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-从收件人那里获取目标信息，并让 Lo CloudWatch gs 知道它已准备好将其日志事件发送到指定目标。在本节其余部分的步骤中，显示日志数据发送者的虚构 AWS 账号为 111111111111。
- 日志数据接收者-设置封装 Kinesis Data Streams 流的目标，并 CloudWatch 让日志知道接收者想要接收日志数据。接收者随后与发送者共享与此目标有关的信息。在本节其余部分的步骤中，显示日志数据接收者的虚构 AWS 账号为222222222222。

本节中的示例使用带有 Amazon S3 存储空间的 Firehose 传输流。您也可以使用不同的设置来设置 Firehose 传送流。有关更多信息，请参阅创建 Firehose 交付流。

> **Note**
>
> 日志组和目标必须位于同一 AWS 区域。但是，目标指向的 AWS 资源可位于不同的区域。

> **Note**
>
> 支持同一账户的 Firehose 订阅过滤器和跨区域传送流。

主题

- 第 1 步：创建 Firehose 传送流
- 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

第 1 步：创建 Firehose 传送流

> **Important**
>
> 在完成以下步骤之前，您必须使用访问策略，这样 Firehose 才能访问您的 Amazon S3 存储桶。有关更多信息，请参阅《亚马逊数据 Firehose 开发者指南》中的控制访问权限。
> 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。

创建用作目标的 Firehose 传送流

1. 创建 Amazon S3 存储桶：

```
aws s3api create-bucket --bucket firehose-test-bucket1 --create-bucket-
configuration LocationConstraint=us-east-1
```

2. 创建 IAM 角色以授予 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. 创建权限策略来定义 Firehose 可以在您的账户中执行的操作。

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

   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. 输入以下命令来创建 Firehose 传送流。*my-bucket-arn*使用正确的部署值替换*my-role-arn*和。

```
aws firehose create-delivery-stream \
   --delivery-stream-name 'my-delivery-stream' \
   --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::firehose-test-bucket1"}'
```

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。

创建目标后，CloudWatch Logs 会代表收件人账户向目标发送一条测试消息。当订阅筛选器稍后处于活动状态时，Logs 会代表源账户将 CloudWatch 日志事件发送到目标。

创建目标

1. 等到你在中创建的 Firehose 直播第 1 步：创建 Firehose 传送流变为激活状态。您可以使用以下命令来检查StreamDescription。 StreamStatus财产。

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
```

此外，还要注意DeliveryStreamDescription。 DeliveryStreamARN 值，因为您需要在以后的步骤中使用它。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:::firehose-test-bucket1",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    }
                },
                "ExtendedS3DestinationDescription": {
                    "RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
                    "BucketARN": "arn:aws:s3:::firehose-test-bucket1",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    },
                    "S3BackupMode": "Disabled"
                }
            }
```

```
        ],
        "HasMoreDestinations": false
    }
}
```

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，创建 IAM 角色将授予 CloudWatch Logs 将数据放入您的 Firehose 流的权限。首先，你需要在文件 ~/ TrustPolicyFor cwl .json 中创建信任策略。使用文本编辑器创建此策略。有关 CloudWatch 日志终端节点的更多信息，请参阅 Amazon CloudWatch Logs 终端节点和配额。

   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": "AROAR3BXASEKYJYWF243H",
        "Arn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
        "CreateDate": "2021-02-02T08:10:43+00:00",
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.region.amazonaws.com"
                },
                "Action": "sts:AssumeRole",
                "Condition": {
                    "StringLike": {
                        "aws:SourceArn": [
                            "arn:aws:logs:region:sourceAccountId:*",
                            "arn:aws:logs:region:recipientAccountId:*"
                        ]
                    }
                }
            }
        }
    }
}
```

4. 创建权限策略以定义 CloudWatch 日志可以对您的账户执行哪些操作。首先，使用文本编辑器在文件 ~/ PermissionsFor cwl .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 传输流处于活动状态并且您已创建 IAM 角色之后，您可以创建 CloudWatch 日志目标。

   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入 ~/ AccessPolicy .json 文件中：

```
{
  "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。跨账户用户将使用该PutSubscriptionFilter操作将日志事件发送到目标：

```
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 AWS 跨账户策略评估逻辑，要访问任何跨账户资源（例如用作订阅过滤器目标的 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 AWS 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*RoleName*或表示的值*UserName*。

2. 登录发送账户并使用您在步骤 1 AWS Management Console 中输入的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。下面的示例策略显示了建议的权限。

   以下策略仅允许在单个 AWS 账户（账户）中对任何目标资源创建订阅筛选器123456789012：

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

   以下策略仅允许在单个 AWS 账户（账户）sampleDestination中命名的特定目标资源上创建订阅筛选器123456789012：

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

```
    ]
}
```

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，过滤器与包含 AWS CloudTrail 事件的日志组相关联，因此通过 “根” AWS 凭据记录的每个活动都将传送到您之前创建的目标。有关如何向 CloudWatch 日志发送 AWS CloudTrail 事件的更多信息，请参阅AWS CloudTrail 用户指南中的向 CloudWatch 日志发送 CloudTrail事件。

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 AWS 区域。但是，目标可以指向位于不同区域的 AWS 资源，例如Firehose流。

验证录入事件流

创建订阅过滤器后，Lo CloudWatch gs 会将所有与过滤器模式匹配的传入日志事件转发到 Firehose 传输流。根据在 Firehose 传输流上设置的时间缓冲间隔，数据开始出现在您的 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 'firehose-test-bucket1'
```

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 'firehose-test-bucket1' --key '2021/02/02/08/my-delivery-stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba' testfile.gz
```

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。您可以将目标上的PutDestinationPolicy操作与新的访问策略结合使用。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. 获取当前与目标 TestDest ination 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。将此政策放入 ~/ NewAccessPolicy .json 文件中：

```
{
  "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相关联：

```
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 的跨账户跨区域账户级别订阅

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：

- 日志数据发送者-从收件人那里获取目标信息，并让 Lo CloudWatch gs 知道它已准备好将其日志事件发送到指定目标。在本节其余部分的步骤中，显示日志数据发送者的虚构 AWS 账号为 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-设置封装 Kinesis Data Streams 流的目标，并 CloudWatch 让日志知道接收者想要接收日志数据。接收者随后与发送者共享与此目标有关的信息。在本节其余部分的步骤中，显示日志数据接收者的虚构 AWS 账号为 999999999999。

要开始接收来自跨账户用户的日志事件，日志数据接收者首先创建 CloudWatch 日志目标。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 Feed 目标的 AWS 资源的亚马逊资源名称 (ARN)。

角色 ARN

一个 AWS Identity and Access Management (IAM) 角色，它向 CloudWatch Logs 授予将数据放入所选数据流的必要权限。

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 AWS 区域。但是，目标指向的 AWS 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
- 第 3 步：创建账户级订阅筛选器策略
- 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，日志数据接收者帐户的 AWS 帐户 ID 为 999999999999，而日志数据发送者 AWS 帐户 ID 为 111111111111。

此示例使用 RecipientStream名为的 Kinesis Data Streams 流创建一个目标，以及一个 CloudWatch 允许日志向其写入数据的角色。

创建目标后， CloudWatch Logs 会代表收件人账户向目标发送一条测试消息。当订阅筛选器稍后处于活动状态时，Logs 会代表源账户将 CloudWatch 日志事件发送到目标。

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。你可以使用 a ws kinesis describe-stream 命令来检查。StreamDescription StreamStatus财产。此外，请记下 StreamDescription.streamArn 的值，因为您稍后会将其传递给 Logs： CloudWatch

```
aws kinesis describe-stream --stream-name "RecipientStream"
{
  "StreamDescription": {
    "StreamStatus": "ACTIVE",
    "StreamName": "RecipientStream",
    "StreamARN": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "340282366920938463463374607431768EXAMPLE",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49551135218688818456679503831981458784591352702 18EXAMPLE"
        }
      }
    ]
  }
}
```

您的流可能需要一两分钟才会以活动状态显示。

3. 创建 IAM 角色以授予 CloudWatch Logs 将数据放入您的直播的权限。首先，你需要在文件 ~/TrustPolicyFor cwl .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。请记下返回的 Role.Arn 值，因为它稍后也会传递给 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": "AAOIIAH450GAB4HC5F431",
        "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 日志可以对您的账户执行哪些操作。首先，使用文本编辑器在文件 ~/ PermissionsFor cwl .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 a w s 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. 在直播处于活动状态并且您已创建 IAM 角色之后，您可以创建 CloudWatch 日志目标。

   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。记下有效载荷中返回的：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 action，并向发件人账户授予访问目标的权限。

该策略向发送日志的 AWS 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。

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

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。

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

```
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111111111111"
      },
      "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
      "Resource" :
 "arn:aws:logs:region:999999999999:destination:testDestination"
    }
  ]
}
```

c. 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
```

####### AWS ###### 111111111111 ##### ARN arn: aws: logs: region: 999999999999: Destinationation: testDestination: te ***PutSubscriptionFilter****stDestination ######*任何其他用户试图拨 PutSubscriptionFilter 打该目的地的电话都将被拒绝。

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，如果您使用 AWS Organizations 的是跨账户权限，请按照中的步骤 2：（仅在使用企业时）创建 IAM 角色步骤操作。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 第 3 步：创建账户级订阅筛选器策略。

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 第 3 步：创建账户级订阅筛选器策略。

本节中的步骤创建一个 IAM 角色，该角色 CloudWatch 可以假设和验证发件人账户是否有权针对收件人目的地创建订阅筛选条件。

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 AWS 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 可以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 第 3 步：创建账户级订阅筛选器策略。

第 3 步：创建账户级订阅筛选器策略

在创建目标后，日志数据接收者账户可与其他 AWS 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账户级别的订阅筛选策略。该筛选器与发件人账户关联，111111111111因此每个符合筛选条件和选择标准的日志事件都会传送到您之前创建的目的地。该目的地封装了一个名为 “” 的流。RecipientStream

该selection-criteria字段是可选的，但对于从订阅筛选器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，请参阅日志递归防护。目前，NOT IN是唯一支持的运算符selection-criteria。

```
aws logs put-account-policy \
    --policy-name "CrossAccountStreamsExamplePolicy" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document
 '{"DestinationArn":"arn:aws:logs:region:999999999999:destination:testDestination",
 "FilterPattern": "", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

发件人账户的日志组和目标必须位于同一 AWS 区域。但是，目标可以指向位于不同区域的 AWS 资源，例如 Kinesis Data Streams 流。

验证日志事件流

创建账户级订阅筛选策略后，Lo CloudWatch gs 会将所有符合过滤模式和选择标准的传入日志事件转发到封装在名为 “” 的目标流中的流。RecipientStream目标所有者可以通过使用 aws kinesis get-shard-iterator 命令获取 Kinesis Data Streams 分片，然后使用 aw s kinesis get-records 命令获取一些 Kinesis Data Streams 记录来验证是否发生了这种情况：

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

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

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

> **Note**
>
> 在 Kinesis Data Streams 开始返回数据之前，你可能需要重新运行该`get-records`命令几次。

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": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        }
    ]
}
```

数据结构中的关键元素如下：

messageType

数据消息将使用“DATA_MESSAGE”类型。有时， CloudWatch 日志可能会发出“CONTROL_MESSAGE” 类型的 Kinesis Data Streams 记录，主要用于检查目标是否可达。

owner

原始日志数据的 AWS 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。

策略级别

政策的执行级别。“账户级别_POLICY” 是policyLevel针对账户级别的订阅筛选器策略。

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. 获取当前与目标 TestDest ination 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。将此政策放入 ~/ NewAccessPolicy .json 文件中：

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

3. 调用PutDestinationPolicy将 NewAccessPolicy.json 文件中定义的策略与目标关联起来：

```
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 启用来自 AWS 服务的日志记录 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。此更新添加了一个 IAM 角色，该角色 CloudWatch 可以假设并验证发件人账户有权向收件人账户发送日志。

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 可以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器策略。

```
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。

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

2. 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 AWS 账户 ID 的访问策略，请包括 force 参数。

> **Warning**
>
> 如果您正在处理中列出的 AWS 服务发送的日志启用来自 AWS 服务的日志记录，则在执行此步骤之前，必须先更新所有发件人账户中的订阅筛选器，如中所述步骤 1：更新订阅筛选条件。

```
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别订阅

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-从收件人那里获取目标信息，并让 Lo CloudWatch gs 知道它已准备好将其日志事件发送到指定目标。在本节其余部分的步骤中，显示日志数据发送者的虚构 AWS 账号为 111111111111。
- 日志数据接收者-设置封装 Kinesis Data Streams 流的目标，并 CloudWatch 让日志知道接收者想要接收日志数据。接收者随后与发送者共享与此目标有关的信息。在本节其余部分的步骤中，显示日志数据接收者的虚构 AWS 账号为222222222222。

本节中的示例使用带有 Amazon S3 存储空间的 Firehose 传输流。您也可以使用不同的设置来设置 Firehose 传送流。有关更多信息，请参阅创建 Firehose 交付流。

> **Note**
>
> 日志组和目标必须位于同一 AWS 区域。但是，目标指向的 AWS 资源可位于不同的区域。

> **Note**
>
> 支持同一账户的 Firehose 订阅过滤器和跨区域传送流。

主题

- 第 1 步：创建 Firehose 传送流
- 步骤 2：创建目标
- 第 3 步：创建账户级订阅筛选器策略
- 验证录入事件流
- 在运行时修改目标成员资格

第 1 步：创建 Firehose 传送流

> **Important**
>
> 在完成以下步骤之前，您必须使用访问策略，这样 Firehose 才能访问您的 Amazon S3 存储桶。有关更多信息，请参阅《亚马逊数据 Firehose 开发者指南》中的控制访问权限。
> 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。

创建用作目标的 Firehose 传送流

1. 创建 Amazon S3 存储桶：

```
aws s3api create-bucket --bucket firehose-test-bucket1 --create-bucket-
configuration LocationConstraint=us-east-1
```

2. 创建 IAM 角色以授予 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. 创建权限策略来定义 Firehose 可以在您的账户中执行的操作。

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

```
    }]
}
```

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. 输入以下命令来创建 Firehose 传送流。*my-bucket-arn*使用正确的部署值替换*my-role-arn*和。

```
aws firehose create-delivery-stream \
   --delivery-stream-name 'my-delivery-stream' \
   --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::firehose-test-bucket1"}'
```

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。

创建目标后，CloudWatch Logs 会代表收件人账户向目标发送一条测试消息。当订阅筛选器稍后处于活动状态时，Logs 会代表源账户将 CloudWatch 日志事件发送到目标。

创建目标

1. 等到你在中创建的 Firehose 直播第 1 步：创建 Firehose 传送流变为激活状态。您可以使用以下命令来检查StreamDescription。 StreamStatus财产。

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
```

此外，还要注意DeliveryStreamDescription。 DeliveryStreamARN 值，因为您需要在以后的步骤中使用它。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:::firehose-test-bucket1",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    }
                },
                "ExtendedS3DestinationDescription": {
                    "RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
                    "BucketARN": "arn:aws:s3:::firehose-test-bucket1",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
```

```
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    },
                    "S3BackupMode": "Disabled"
                }
            }
        ],
        "HasMoreDestinations": false
    }
}
```

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，创建 IAM 角色将授予 CloudWatch Logs 将数据放入您的 Firehose 流的权限。首先，你需要在文件 ~/ TrustPolicyFor cwl .json 中创建信任策略。使用文本编辑器创建此策略。有关 CloudWatch 日志终端节点的更多信息，请参阅 Amazon CloudWatch Logs 终端节点和配额。

   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": "AROAR3BXASEKYJYWF243H",
        "Arn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
        "CreateDate": "2023-02-02T08:10:43+00:00",
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.amazonaws.com"
                },
                "Action": "sts:AssumeRole",
                "Condition": {
                    "StringLike": {
                        "aws:SourceArn": [
                            "arn:aws:logs:region:sourceAccountId:*",
                            "arn:aws:logs:region:recipientAccountId:*"
                        ]
                    }
                }
            }
        }
    }
}
```

4. 创建权限策略以定义 CloudWatch 日志可以对您的账户执行哪些操作。首先，使用文本编辑器在文件 ~/ PermissionsFor cwl .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 传输流处于活动状态并且您已创建 IAM 角色之后，您可以创建 CloudWatch 日志目标。

   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入`~/AccessPolicy.json`文件中：

```
{
  "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操作将日志事件发送到目标。

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

第 3 步：创建账户级订阅筛选器策略

切换到发送账户，在此示例中为 111111111111。现在，您将在发送账户中创建账户级别的订阅筛选策略。在此示例中，过滤器会将除两个日志组之外的所有日志组ERROR中包含该字符串的每个日志事件传送到您之前创建的目标。

```
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 AWS 区域。但是，目标可以指向位于不同区域的 AWS 资源，例如Firehose流。

验证录入事件流

创建订阅过滤器后，Lo CloudWatch gs 会将所有符合筛选模式和选择标准的传入日志事件转发到 Firehose 传输流。根据在 Firehose 传输流上设置的时间缓冲间隔，数据开始出现在您的 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 'firehose-test-bucket1'
```

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 'firehose-test-bucket1' --key '2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba' testfile.gz
```

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。您可以将PutDestinationPolicy和PutAccountPolicy操作与新的访问策略一起用于您的目的地。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. 获取当前与目标 TestDest ination 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}\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。将此政策放入 ~/ NewAccessPolicy .json 文件中：

```
{
  "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相关联：

```
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。在中 AWS，跨服务模仿可能会导致混乱的副手问题。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。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。为了防止这种情况，我们 AWS 提供了一些工具，帮助您保护所有服务的数据，这些服务委托人已被授予对您账户中资源的访问权限。

我们建议在资源策略中使用aws:SourceArnaws:SourceAccountaws:SourceOrgID、、和aws:SourceOrgPaths全局条件上下文密钥来限制为资源提供其他服务的权限。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 AWS Organizations 路径内账户中的任何资源与跨服务使用相关联。有关使用和理解路径的更多信息，请参阅了解 AWS 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。

记录的策略用于授予访问 CloudWatch 日志的权限，以便将数据写入 Kinesis Data Streams 和 Firehose步骤 1：创建目标，步骤 2：创建目标并展示了如何使用 awsSourceArn : 全局条件上下文密钥来帮助防止混乱的副手问题。

日志递归防护

订阅过滤器存在导致无限日志递归的风险，如果不加以防止，则可能导致 CloudWatch 日志和目标中的摄取账单大幅增加。当订阅筛选器与接收订阅交付工作流程产生的日志事件的日志组关联时，可能会发生这种情况。提取到日志组中的日志将传送到目标，从而导致日志组摄取更多日志，然后这些日志将再次转发到目标，从而形成递归循环。

例如，考虑一个目标为 Firehose 的订阅筛选器，它将日志事件传送到 Amazon S3。此外，还有一个 Lambda 函数，用于处理传送到 Amazon S3 的新事件并自行生成一些日志。如果将订阅筛选器应用于 Lambda 函数的日志组，则该函数生成的日志事件将被转发到目标位置的 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转发到 Firehose 和 Amazon S3，从而导致再次调用该函数等。这将以无限循环的方式发生，导致日志提取、Firehose 和 Amazon S3 的账单意外增加。

如果 Lambda 函数附加到启用日志流日志的 VPC，则该 VPC 的日志组也可能导致日志递归。CloudWatch

我们建议您不要将订阅过滤器应用于订阅交付工作流程中的日志组。对于账户级订阅过滤器，请使用 PutAccountPolicy API 中的selectionCriteria参数将这些日志组排除在策略之外。

排除日志组时，请考虑以下生成日志的 AWS 服务，这些服务可能是您的订阅交付工作流程的一部分：

- 带有 Fargate 的 Amazon EC2
- Lambda
- AWS Step Function
- 为日志启用的 Amazon VPC 流 CloudWatch 日志

> **Note**
>
> Lambda 目标的日志组生成的日志事件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 Logs Insights 查询语言查询 CloudWatch 日志组的信息，请参阅CloudWatch 日志见解查询语法。

借助 CloudWatch 日志，您可以使用指标筛选器将日志数据转换为可操作的指标，使用订阅过滤器将日志事件路由到其他 AWS 服务，使用筛选日志事件来搜索日志事件，使用L ive Tail 在采集日志时以交互方式实时查看日志。

筛选条件模式构成了指标筛选条件、订阅筛选条件、筛选日志事件和 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。你可以在 CloudWatch 控制台中测试过滤器模式。

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**
>
> 配额
> 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。
- ?：匹配前一个术语的零个或多个实例。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”。

  > **Note**
  >
  > 字词是单个字符或重复字符类，其使用以下运算符之一：?、*、+ 或 {n,m}。

- \：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。

- *：匹配前一个术语的零个或多个实例。例如， %ab*c% 可以匹配“ac”、“abc”和“abbbc”；%ab[0-9]*% 可以匹配“ab”、“ab0”和“ab129”。
- +：匹配前一个术语的一个或多个实例。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“?”与其他筛选条件模式组合，问号（“?”）将被忽略。
> 例如，以下筛选条件模式匹配所有包含词语 REQUEST 的事件，但带有问号（“?”）的筛选条件将被忽略且无效。
>
> ```
> ?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。
  > 配额
  > 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，显示了可以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。
> 配额
> 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 ("<=")。

- 数字

  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，显示了可以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**
>
> 配额

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**
>
> 配额
> 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条件模式，其返回第一个单词为 ERROR 的日志事件，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]
```

启用来自 AWS 服务的日志记录

虽然许多服务仅向日志发布日 CloudWatch 志，但有些 AWS 服务可以将日志直接发布到亚马逊简单存储服务或 Amazon Data Firehose。如果您对日志的主要要求是在其中一项服务中进行存储或处理，则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，而无需进行额外设置。

即使日志直接发布到 Amazon S3 或 Firehose，也需要付费。有关更多信息，请参阅 Amazon P ricing 中 “日志” 选项卡上的 “销售日志” CloudWatch 。

一些 AWS 服务使用通用基础架构来发送日志。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 AWS 向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。

日志类型	CloudWatch Logs	Amazon S3	Firehose
Amazon API Gateway 访问日志	支持的 [V1 权限]		
AWS AppSync 日志	支持		
Amazon Aurora MySQL 日志	支持		
Amazon Bedrock 知识库日志	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	支持的 [V1 权限]		
CloudFront: 访问日志		支持的 [V1 权限]	
AWS CloudHSM 审核日志	支持		
CloudWatch 显然是评估事件日志	支持的 [V1 权限]	支持的 [V1 权限]	

日志类型	CloudWatch Logs	Amazon S3	Firehose
CloudWatch 互联网监视器日志		支持的 [V1 权限]	
CloudTrail 日志	支持		
AWS CodeBuild 日志	支持		
Amazon CodeWhisperer 事件日志	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]
Amazon Cognito 日志	支持的 [V1 权限]		
Amazon Connect 日志	支持		
AWS DataSync 日志	支持		
Amazon ElastiCache for Redis 日志	支持的 [V1 权限]		支持的 [V1 权限]
AWS Elastic Beanstalk 日志	支持		
Amazon Elastic Container Service 日志	支持		
Amazon Elastic Kubernetes Service 控制面板日志	支持		
Amazon EventBridge 管道记录	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]
AWS Fargate 日志	支持		
AWS Fault Injection Service 实验日志		支持的 [V1 权限]	
Amazon FinSpace	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]

日志类型	CloudWatch Logs	Amazon S3	Firehose
AWS Global Accelerator 流日志		支持的 [V1 权限]	
AWS Glue 作业日志	支持		
IAM 身份中心错误日志	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	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]
AWS IoT 日志	支持		
AWS IoT FleetWise 日志	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]
AWS Lambda 日志	支持		
Amazon Macie 日志	支持		
AWS Mainframe Modernization	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	支持的 [V1 权限]		
Amazon MSK 代理日志	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]
Amazon MSK Connect 日志	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]
Amazon MQ 一般日志和审计日志	支持		
AWS Network 防火墙日志	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]

日志类型	CloudWatch Logs	Amazon S3	Firehose
Network Load Balancer 访问日志		支持的 [V1 权限]	
OpenSearch 日志	支持		
Amazon OpenSearch 服务摄取日志	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]
AWS OpsWorks 日志	支持		
亚马逊关系数据库 ServicePostgre SQL 日志	支持		
AWS RoboMaker 日志	支持		
Amazon Route 53 公有 DNS 查询日志	支持		
Amazon Route 53 Resolver 查询日志	支持的 [V1 权限]	支持的 [V1 权限]	
亚马逊 SageMaker 活动	支持的 [V1 权限]		
亚马逊 SageMaker 员工活动	支持的 [V1 权限]		
AWS 站点到站点 VPN 日志	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]
Amazon Simple Notification Service 日志	支持		
Amazon Simple Notification Service 数据保护策略日志	支持		
EC2 Spot 实例数据源文件		支持的 [V1 权限]	

日志类型	CloudWatch Logs	Amazon S3	Firehose
AWS Step Functions 快速工作流程和标准工作流程日志	支持的 [V1 权限]		
Storage Gateway 审计日志和运行状况日志	支持的 [V1 权限]		
AWS Transfer Family 日志	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]
AWS Verified Access 日志	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]
Amazon Virtual Private Cloud 流日志	支持	支持的 [V1 权限]	支持的 [V1 权限]
Amazon VPC Lattice 访问日志	支持的 [V1 权限]	支持的 [V1 权限]	支持的 [V1 权限]
AWS WAF 日志	支持的 [V1 权限]	支持的 [V1 权限]	支持
Amazon WorkMail 日志	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

有些 AWS 服务使用通用基础设施将其日志发送到日 CloudWatch 志、Amazon S3 或 Firehose。要启用下表中列出的 AWS 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，必须向 AWS 授予权限才能发送日志。 AWS 可以在设置日志时自动创建这些权限，也可以在设置日志之前先自己创建这些权限。对于跨账户交付，您必须自己手动创建权限策略。

如果您选择在您或您的组织中的某人首次设置日志发送时 AWS 自动设置必要的权限和资源策略，则设置发送日志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。

下表汇总了适用本节中信息的日志类型及日志目标。

以下各部分提供了每个目标的更多详细信息。

发送到日志的 CloudWatch 日志

> **Important**
>
> 在将以下列表中的日志类型设置为发送到 Lo CloudWatch gs 时，如果需要，可以 AWS 创建或更改与接收日志的日志组关联的资源策略。继续阅读本节，查看详细信息。

当将上一节表中列出的日志类型发送到 Log CloudWatch s 时，本节适用：

用户权限

为了能够设置首次向 Logs 发送任何此类 CloudWatch 日志，您必须使用具有以下权限的账户登录。

- `logs:CreateLogDelivery`
- `logs:PutResourcePolicy`
- `logs:DescribeResourcePolicies`
- `logs:DescribeLogGroups`

> **Note**
>
> 指定`logs:DescribeLogGroups`、`logs:DescribeResourcePolicies`或`logs:PutResourcePolicy`权限时，请务必将其`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:*"`

如果这些类型的日志中的任何一种已发送到日志中的某个 CloudWatch 日志组，则要设置向同一日志组发送另一类此类日志，您只需要该`logs:CreateLogDelivery`权限即可。

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，并且设置日志记录的用户拥有该日志组的`logs:PutResourcePolicy`、`logs:DescribeResourcePolicies`和`logs:DescribeLogGroups`权限，则在您开始将日志发送到 CloudWatch Logs 时， AWS 会自动为其创建以下策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite20150319",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "delivery.logs.amazonaws.com"
        ]
      },
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:0123456789:log-group:my-log-group:log-stream:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": ["0123456789"]
        },
        "ArnLike": {
          "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:*"]
        }
      }
    }
  ]
}
```

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。

日志组资源策略大小限制注意事项

这些服务必须在资源策略中列出要向其发送日志的每个日志组，并且 CloudWatch 日志资源策略限制为 5120 个字符。向大量日志组发送日志的服务可能会遇到此限制。

为了缓解这种情况， CloudWatch Logs 会监控发送日志的服务所使用的资源策略的大小，当它检测到策略接近 5120 个字符的大小限制时， CloudWatch 日志会自动在该服务的资源策略/aws/

vendedlogs/*中启用。之后，您可以开始将名称以 /aws/vendedlogs/ 开头的日志组作为这些服务所发送的日志的目标。

发送到 Amazon S3 的日志

将日志设置为发送到 Amazon S3 时，如果需要，可以 AWS 创建或更改与接收日志的 S3 存储桶关联的资源策略。

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 使用的权限模型与其他向 S3 发送已售日志的服务不同。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 访问日志使用相同的 S3 存储桶和另一个日志源，则在存储桶上启用 ACL CloudFront 也会向使用此存储桶的所有其他日志源授予权限。

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，并且设置日志记录的用户拥有该存储桶的`S3:GetBucketPolicy`和`S3:PutBucketPolicy`权限，则在您开始向 Amazon S3 发送日志时 AWS 会自动为其创建以下策略。

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

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**
>
> 在某些情况下， AWS CloudTrail 如果未授予s3:ListBucket权限，则可能会在中看到AccessDenied错误delivery.logs.amazonaws.com。为避免 CloudTrail 日志中出现这些错误，您必须向授予s3:ListBucket权限，delivery.logs.amazonaws.com并且必须包含在前面的存储桶策略中设置的s3:GetBucketAcl权限中显示的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

您可以通过使用 Amazon S3 托管密钥启用服务器端加密 (SSE-S3) 或使用存储在 (SSE-KMS) 中的密钥启用服务器端加密 (SSE-KMS) 来保护 Amazon S3 存储桶中的 AWS Key Management Service 数据。 AWS KMS 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**
>
> 如果您选择 SSE-KMS，则必须使用客户托管密钥，因为这种情况不支持使用 AWS 托管密钥。如果您使用 AWS 托管密钥设置加密，则日志将以不可读的格式传送。

当您使用客户托管 AWS KMS 密钥时，您可以在启用存储桶加密时指定客户托管密钥的 Amazon 资源名称 (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。

如果您选择 SSE-KMS，则必须使用客户托管密钥，因为这种情况不支持使用 AWS 托管密钥。当您使用客户托管 AWS KMS 密钥时，您可以在启用存储桶加密时指定客户托管密钥的 Amazon 资源名称

(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。

已发送到 Firehose 的日志

当将上一节表格中列出的日志类型发送到 Firehose 时，本节适用：

用户权限

要设置首次向 Firehose 发送任何此类日志，您必须使用具有以下权限的账户登录。

- logs:CreateLogDelivery
- firehose:TagDeliveryStream
- iam:CreateServiceLinkedRole

如果这些类型的日志中的任何一个已经发送到 Firehose，那么要设置向 Firehose 发送另一种此类日志，您只需要拥有和权限。`logs:CreateLogDelivery` `firehose:TagDeliveryStream`

用于权限的 IAM 角色

由于 Firehose 不使用资源策略， AWS 因此在设置要发送到 Firehose 的日志时会使用 IAM 角色。AWS 创建名AWSServiceRoleForLogDelivery为的服务相关角色。此服务相关角色包括以下权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch",
                "firehose:ListTagsForDeliveryStream"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/LogDeliveryEnabled": "true"
                }
            },
            "Effect": "Allow"
        }
    ]
}
```

此服务相关角色授予标签设置为的所有 Firehose 传送流`LogDeliveryEnabled`的权限。`true` AWS 在设置日志记录时，将此标签提供给目标传送流。

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：

```
{
  "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 AWS 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到以下一个或多个目标的日志传输： CloudWatch 日志、Amazon S3 或 Firehose。

工作日志交付由三个元素组成：

- ADeliverySource，它是一个逻辑对象，代表实际发送日志的资源。
- ADeliveryDestination，它是一个逻辑对象，代表实际的交付目的地。
- ADelivery，它将传送源与传送目标连接起来

要在支持的 AWS 服务和目标之间配置日志传输，必须执行以下操作：

- 使用创建交付来源PutDeliverySource。
- 使用创建配送目的地PutDeliveryDestination。
- 如果您要跨账户传送日志，则必须在目标账户 PutDeliveryDestinationPolicy中使用向目标分配 IAM 策略。此策略授权创建从账户 A 中的交付源到账户 B 中的传送目标的交付。对于跨账户交付，您必须自己手动创建权限策略。
- 通过使用将一个配送来源和一个配送目的地精确配对来创建配送 CreateDelivery。

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，您有责任移除日志传输资源。为此，请按照以下步骤操作。
>
> 1. Delivery使用DeleteDelivery操作删除。
> 2. DeliverySource使用DeleteDeliverySource操作删除。
> 3. 如果与您刚刚删除的DeliveryDestinationDeliverySource关联仅用于此特定用途DeliverySource，则可以使用DeleteDeliveryDestinations操作将其删除。

目录

- 发送到日志的 CloudWatch 日志
- 发送到 Amazon S3 的日志
  - Amazon S3 存储桶服务器端加密
- 已发送到 Firehose 的日志
- 特定于服务的权限
- 控制台专属权限

发送到日志的 CloudWatch 日志

用户权限

要启用向日志发送 CloudWatch 日志，您必须使用以下权限登录。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:GetDelivery",
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
                "logs:GetDeliveryDestinationPolicy",
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:delivery:*",
                "arn:aws:logs:region:account-id:delivery-source:*",
                "arn:aws:logs:region:account-id:delivery-destination:*"
            ]
        },
```

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

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，并且设置日志记录的用户拥有该日志组的logs:PutResourcePolicylogs:DescribeResourcePolicies、和logs:DescribeLogGroups权限，则在您开始将日志发送到 CloudWatch Logs 时， AWS 会自动为其创建以下策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite20150319",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "delivery.logs.amazonaws.com"
        ]
      },
```

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

日志组资源策略大小限制注意事项

这些服务必须在资源策略中列出要向其发送日志的每个日志组，并且 CloudWatch 日志资源策略限制为 5120 个字符。将日志发送到大量日志组的服务可能会遇到此限制。

为了缓解这种情况， CloudWatch Logs 会监控发送日志的服务所使用的资源策略的大小，当它检测到策略接近 5120 个字符的大小限制时， CloudWatch 日志会自动在该服务的资源策略/aws/vendedlogs/*中启用。之后，您可以开始将名称以 /aws/vendedlogs/ 开头的日志组作为这些服务所发送的日志的目标。

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。

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

```
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
                "logs:GetDeliveryDestinationPolicy",
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:delivery:*",
                "arn:aws:logs:region:account-id:delivery-source:*",
                "arn:aws:logs:region:account-id:delivery-destination:*"
            ]
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyS3",
            "Effect": "Allow",
            "Action": [
                "s3:PutBucketPolicy",
                "s3:GetBucketPolicy"
            ],
            "Resource": "arn:aws:s3:::bucket-name"
        }
    ]
}
```

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，并且设置日志记录的用户拥有该存储桶的`S3:GetBucketPolicy`和`S3:PutBucketPolicy`权限，则在您开始向 Amazon S3 发送日志时 AWS 会自动为其创建以下策略。

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

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**
>
> 在某些情况下， AWS CloudTrail 如果未授予s3:ListBucket权限，则可能会在中看到AccessDenied错误delivery.logs.amazonaws.com。为避免 CloudTrail 日志中出现这些错误，您必须向授予s3:ListBucket权限，delivery.logs.amazonaws.com并且必须包含在前面的存储桶策略中设置的s3:GetBucketAcl权限中显示的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

您可以通过使用 Amazon S3 托管密钥启用服务器端加密 (SSE-S3) 或使用存储在 (SSE-KMS) 中的密钥启用服务器端加密 (SSE-KMS) 来保护 Amazon S3 存储桶中的 AWS Key Management Service 数据。 AWS KMS 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**
>
> 如果您选择 SSE-KMS，则必须使用客户托管密钥，因为这种情况不支持使用 AWS 托管密钥。如果您使用 AWS 托管密钥设置加密，则日志将以不可读的格式传送。

当您使用客户托管 AWS KMS 密钥时，您可以在启用存储桶加密时指定客户托管密钥的 Amazon 资源名称 (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。

如果您选择 SSE-KMS，则必须使用客户托管密钥，因为这种情况不支持使用 AWS 托管密钥。当您使用客户托管 AWS KMS 密钥时，您可以在启用存储桶加密时指定客户托管密钥的 Amazon 资源名称

(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。

已发送到 Firehose 的日志

用户权限

要启用向 Firehose 发送日志，您必须使用以下权限登录。

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

```
            "Action": [
                "logs:GetDelivery",
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
                "logs:GetDeliveryDestinationPolicy",
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:delivery:*",
                "arn:aws:logs:region:account-id:delivery-source:*",
                "arn:aws:logs:region:account-id:delivery-destination:*"
            ]
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyFH",
            "Effect": "Allow",
            "Action": [
                "firehose:TagDeliveryStream"
            ],
            "Resource": [
                "arn:aws:firehose:region:account-id:deliverystream/*"
            ]
        },
        {
            "Sid": "CreateServiceLinkedRole",
            "Effect": "Allow",
            "Action": [
```

```
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::account-id:role/aws-service-role/
delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery"
        }
    ]
}
```

用于资源权限的 IAM 角色

由于 Firehose 不使用资源策略，AWS 因此在设置要发送到 Firehose 的日志时会使用 IAM 角色。AWS 创建名AWSServiceRoleForLogDelivery为的服务相关角色。此服务相关角色包括以下权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch",
                "firehose:ListTagsForDeliveryStream"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/LogDeliveryEnabled": "true"
                }
            },
            "Effect": "Allow"
        }
    ]
}
```

此服务相关角色授予标签设置为的所有 Firehose 传送流LogDeliveryEnabled的权限。`true` AWS 在设置日志记录时，将此标签提供给目标传送流。

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：

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

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

特定于服务的权限

除了前几节中列出的特定于目的地的权限外，某些服务还需要明确授权，允许客户从其资源发送日志，以此作为额外的安全层。它授权对该服务中供应日志的资源AllowVendedLogDeliveryForResource执行操作。对于这些服务，请使用以下策略并将*service*替换为相应的值。有关这些字段的服务特定值，请参阅这些服务的文档页面以获取已售日志。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ServiceLevelAccessForLogDelivery",
            "Effect": "Allow",
            "Action": [
                "service:AllowVendedLogDeliveryForResource"
            ],
            "Resource": "arn:aws:service:region:account-id:resource-type/*"
        }
    ]
}
```

控制台专属权限

除了前几节中列出的权限外，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：

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

```
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:*"
            ]
        },
        {
            "Sid": "AllowLogDeliveryActionsConsoleS3",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Sid": "AllowLogDeliveryActionsConsoleFH",
            "Effect": "Allow",
            "Action": [
                "firehose:ListDeliveryStreams",
                "firehose:DescribeDeliveryStream"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

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。在中 AWS，跨服务模仿可能会导致混乱的副手问题。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， AWS 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:SourceArnaws:SourceAccountaws:SourceOrgID、、和aws:SourceOrgPaths全局条件上下文密钥来限制 CloudWatch Logs 向该资源提供的其他服务的

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 AWS Organizations 路径内账户中的任何资源与跨服务使用相关联。有关使用和理解路径的更多信息，请参阅了解 AWS 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。

本页前几节中的策略显示了如何使用 aws:SourceArn 和 aws:SourceAccount 全局条件上下文密钥，以避免混淆代理人问题出现。

CloudWatch 记录 AWS 托管策略的更新

查看自该服务开始跟踪 CloudWatch 日志 AWS 托管策略更改以来这些更新的详细信息。要获得有关此页面更改的自动提醒，请订阅 “ CloudWatch 日志文档历史记录” 页面上的 RSS feed。

更改	描述	日期
AWSServiceRoleForLogDelivery 服务相关角色策略-更新现有策略	CloudWatch 日志更改了与AWSServiceRoleForLogDelivery服务相关角色关联的 IAM 策略中的权限。更改内容如下： • firehose:ResourceTag/LogDeliveryEnabled": "true" 条件	2021 年 7 月 15 日

更改	描述	日期
	键已更改为 `aws:ResourceTag/LogDeliveryEnabled": "true"` 。	
CloudWatch 日志已开始跟踪更改	CloudWatch 日志开始跟踪其 AWS 托管策略的更改。	2021 年 6 月 10 日

将日志数据导出到 Amazon S3

将日志数据从日志组导出到 Amazon S3 桶，然后使用此数据进行自定义处理和分析，或将其载入到其他系统。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 () 中由 SSE-KMS 加密的 S3 存储桶 AWS Key Management Service AWS KMS
- 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**
>
> 只有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。

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，以便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]+/\x0&/' | 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 “日志见解” 分析 CloudWatch 日志数据 或 使用订阅实时处理日志数据。

内容

- 概念
- 使用控制台将日志数据导出到 Amazon S3
- 使用将日志数据导出到 Amazon S3 AWS CLI
- 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。该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，您使用亚马逊 CloudWatch 控制台将名为的 Amazon Log CloudWatch s 日志组中的所有数据导出`my-log-group`到名为的 Amazon S3 存储桶`my-exported-logs`。

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：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 Logs 创建的存储 CloudWatch 桶。但是，如果要使用现有存储桶，请跳至第 2 步。

> **Note**
>
> S3 存储桶必须与要导出的日志数据位于同一个区域中。 CloudWatch 日志不支持将数据导出到其他区域的 S3 存储桶。

创建 S3 存储桶

1. 打开 Amazon S3 控制台，网址为：https://console.aws.amazon.com/s3/。
2. 如果需要，可以更改区域。从导航栏中，选择您的 CloudWatch 日志所在的区域。
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. 对于区域，选择您的 CloudWatch 日志数据所在的区域。
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：

- 中的用户和群组 AWS IAM Identity Center：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。

- 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》中为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 AWS 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`密钥中的账户 ID 列表将是用户可以从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`。

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

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

4. 选择 Save，将您刚添加的策略设置为存储桶上的访问策略。此策略允许 CloudWatch 日志将日志数据导出到您的 S3 存储桶。存储桶拥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了一个或多个策略，请添加该策略的 CloudWatch 日志访问权限声明。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加密的 S3 存储桶时，才需要执行此步骤。 AWS KMS keys这种加密称为 SSE-KMS。

导出到使用 SSE-KMS 加密的桶

1. 打开 AWS KMS 控制台，网址为 https://console.aws.amazon.com/kms。
2. 要更改 AWS 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。

```
{
    "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. 打开 Amazon S3 控制台，网址为：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 AWS Key Management Service key (SSE-KMS) (Amazon Key Management Service 密钥(SSE-KMS))。
20. 选择从 AWS KMS 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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
3. 在导航窗格中，选择 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）。

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 Logs 创建的存储 CloudWatch 桶。但是，如果要使用现有存储桶，请跳至第 2 步。

> **Note**
>
> S3 存储桶必须与要导出的日志数据位于同一个区域中。 CloudWatch 日志不支持将数据导出到其他区域的 S3 存储桶。

创建 S3 存储桶

1. 打开 Amazon S3 控制台，网址为：https://console.aws.amazon.com/s3/。
2. 如果需要，可以更改区域。从导航栏中，选择您的 CloudWatch 日志所在的区域。
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. 对于区域，选择您的 CloudWatch 日志数据所在的区域。
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 拥有目标账户中目标 Amazon S3 存储桶的`s3:PutObject`权限。

您创建的策略取决于目标存储桶是否使用 AWS KMS 加密。

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 AWS KMS 加密，请将以下策略粘贴到编辑器中。

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

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

如果目标存储桶确实使用了 AWS KMS 加密，请将以下策略粘贴到编辑器中。

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

6. 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：

- 中的用户和群组 AWS IAM Identity Center：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。

- 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》中为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 AWS 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`密钥中的账户 ID 列表将是用户可以从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`。

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

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

4. 选择 Save，将您刚添加的策略设置为存储桶上的访问策略。此策略允许 CloudWatch 日志将日志数据导出到您的 S3 存储桶。存储桶拥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了一个或多个策略，请添加该策略的 CloudWatch 日志访问权限声明。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加密的 S3 存储桶时，才需要执行此步骤。 AWS KMS keys这种加密称为 SSE-KMS。

导出到使用 SSE-KMS 加密的桶

1. 打开 AWS KMS 控制台，网址为 https://console.aws.amazon.com/kms。
2. 要更改 AWS 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。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow CWL Service Principal usage",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.Region.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
```

```
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "account-ARN"
            },
            "Action": [
                "kms:GetKeyPolicy*",
                "kms:PutKeyPolicy*",
                "kms:DescribeKey*",
                "kms:CreateAlias*",
                "kms:ScheduleKeyDeletion*",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM Role Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS":
 "arn:aws:iam::create_export_task_caller_account:role/role_name"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "ARN_OF_KMS_KEY"
        }
    ]
}
```

14. 选择保存更改。
15. 打开 Amazon S3 控制台，网址为：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 AWS Key Management Service key (SSE-KMS) (Amazon Key Management Service 密钥(SSE-KMS))。
20. 选择从 AWS KMS 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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
3. 在导航窗格中，选择 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）。

使用将日志数据导出到 Amazon S3 AWS CLI

在以下示例中，您使用导出任务将名为的 CloudWatch 日志日志组中的所有数据导出`my-log-group`到名为的 Amazon S3 存储桶`my-exported-logs`。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 AWS 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 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 Logs 创建的存储 CloudWatch 桶。但是，如果要使用现有存储桶，请跳至第 2 步。

> **Note**
>
> S3 存储桶必须与要导出的日志数据位于同一个区域中。 CloudWatch 日志不支持将数据导出到其他区域的 S3 存储桶。

使用创建 S3 存储桶 AWS CLI

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：

- 中的用户和群组 AWS IAM Identity Center：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。
- 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》中为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密钥中的账户 ID 列表将是用户可以从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。

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

```
            "Condition": {
              "StringEquals": {
                  "s3:x-amz-acl": "bucket-owner-full-control",
                  "aws:SourceAccount": [
                      "AccountId1",
                      "AccountId2",
                      ...
                  ]
              },
              "ArnLike": {
                      "aws:SourceArn": [
                          "arn:aws:logs:Region:AccountId1:log-group:*",
                          "arn:aws:logs:Region:AccountId2:log-group:*",
                          ...
                      ]
              }
            }
        }
    ]
}
```

2. 使用put-bucket-policy命令将您刚刚添加的策略设置为存储桶的访问策略。此策略允许 CloudWatch 日志将日志数据导出到您的 S3 存储桶。存储桶拥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了一个或多个策略，请添加该策略的 CloudWatch 日志访问权限声明。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加密的 S3 存储桶时，才需要执行此步骤。 AWS KMS keys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。

```
{
    "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",
        "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。

要使用将数据导出到 Amazon S3 AWS CLI

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 Logs 创建的存储 CloudWatch 桶。但是，如果要使用现有存储桶，请跳至第 2 步。

> **Note**
>
> S3 存储桶必须与要导出的日志数据位于同一个区域中。 CloudWatch 日志不支持将数据导出到其他区域的 S3 存储桶。

使用创建 S3 存储桶 AWS CLI

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 策略，以使 CloudWatch 日志拥有目标 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 AWS KMS 加密。如果它不使用 AWS KMS 加密，请创建包含以下内容的策略。

```
{
    "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 AWS KMS 加密，请创建包含以下内容的策略。

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

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：

- 中的用户和群组 AWS IAM Identity Center：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。

- 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》中为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密钥中的账户 ID 列表将是用户可以从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。

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

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

```
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-exported-logs/*",
            "Condition": {
              "StringEquals": {
                  "s3:x-amz-acl": "bucket-owner-full-control"
              }
            }
        }
    ]
}
```

2. 使用put-bucket-policy命令将您刚刚添加的策略设置为存储桶的访问策略。此策略允许 CloudWatch 日志将日志数据导出到您的 S3 存储桶。存储桶拥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了一个或多个策略，请添加该策略的 CloudWatch 日志访问权限声明。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加密的 S3 存储桶时，才需要执行此步骤。 AWS KMS keys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。

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

```
    "Statement": [
        {
            "Sid": "Allow CWL Service Principal usage",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.Region.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "account-ARN"
            },
            "Action": [
                "kms:GetKeyPolicy*",
                "kms:PutKeyPolicy*",
                "kms:DescribeKey*",
                "kms:CreateAlias*",
                "kms:ScheduleKeyDeletion*",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM Role Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS":
 "arn:aws:iam::create_export_task_caller_account:role/role_name"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "ARN_OF_KMS_KEY"
        }
    ]
```

```
}
```

2. 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。

要使用将数据导出到 Amazon S3 AWS CLI

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。

要描述导出任务，请使用 AWS CLI

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 CWLExportUser 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。

要取消导出任务，请使用 AWS CLI

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 日志数据流式传输到 Amazon OpenSearch 服务

您可以配置日志组，使其通过 CloudWatch CloudWatch 日志订阅近乎实时地将其接收到的数据流式传输到您的 Amazon S OpenSearch ervice 集群。有关更多信息，请参阅 使用订阅实时处理日志数据。

> **Note**
>
> 只有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。

根据要流式传输的日志数据量，您可能希望对函数设置函数级别并发执行限制。有关更多信息，请参阅 Lambda 函数扩展。

> **Note**
>
> 将大量 CloudWatch 日志数据流式传输到 OpenSearch 服务可能会导致高额使用费。我们建议您在 AWS Billing and Cost Management 控制台中创建预算。有关更多信息，请参阅通过 AWS Budgets 管理成本。

先决条件

在开始之前，请先创建一个 OpenSearch 服务域。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。您可能需要稍后查看您的 OpenSearch 服务域设置，并根据集群将要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 服务域。

有关 OpenSearch 服务的更多信息，请参阅《亚马逊 OpenSearch 服务开发者指南》。

为日志组订阅 OpenSearch 服务

您可以使用 CloudWatch 控制台为日志组订阅 S OpenSearch ervice。

为日志组订阅 OpenSearch 服务

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。

2. 在导航窗格中，选择 日志组。
3. 选择日志组的名称。
4. 选择操作、订阅筛选条件、创建亚马逊 OpenSearch 服务订阅筛选条件。
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 执行角色，请选择 Lambda 在执行调用时应使用的 IAM 角色。 OpenSearch

   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：

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

   - 如果目标 OpenSearch 服务域使用 VPC 访问权限，则该角色必须附加 AWSLambdaVPCAccessExecutionRole策略。这项由亚马逊管理的策略授予 Lambda 访问客户的 VPC 的权限，从而允许 Lambda 写入到 VPC 中的终端节点。 OpenSearch
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。这样可以确保您只向 OpenSearch 集群发送您感兴趣的数据。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 AWS SDK 的 CloudWatch 日志的代码示例

以下代码示例展示了如何将 CloudWatch 日志与 AWS 软件开发套件 (SDK) 配合使用。

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和跨服务示例的上下文查看操作。

场景是展示如何通过在同一服务中调用多个函数来完成特定任务任务的代码示例。

跨服务示例是指跨多个 AWS 服务工作的示例应用程序。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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 AWS SDK 对 CloudWatch 日志执行的操作
  - AssociateKmsKey与 AWS SDK 或 CLI 配合使用
  - CancelExportTask与 AWS SDK 或 CLI 配合使用
  - CreateExportTask与 AWS SDK 或 CLI 配合使用
  - CreateLogGroup与 AWS SDK 或 CLI 配合使用
  - CreateLogStream与 AWS SDK 或 CLI 配合使用
  - DeleteLogGroup与 AWS SDK 或 CLI 配合使用
  - DeleteSubscriptionFilter与 AWS SDK 或 CLI 配合使用
  - DescribeExportTasks与 AWS SDK 或 CLI 配合使用
  - DescribeLogGroups与 AWS SDK 或 CLI 配合使用
  - DescribeSubscriptionFilters与 AWS SDK 或 CLI 配合使用
  - GetQueryResults与 AWS SDK 或 CLI 配合使用
  - PutSubscriptionFilter与 AWS SDK 或 CLI 配合使用
  - StartLiveTail与 AWS SDK 或 CLI 配合使用
  - StartQuery与 AWS SDK 或 CLI 配合使用
- 使用 AWS SDK 的 CloudWatch 日志场景
  - 使用 CloudWatch 日志运行大型查询
- 使用 AWS SDK 的 CloudWatch 日志的跨服务示例

- 使用计划的事件调用 Lambda 函数

使用 AWS SDK 对 CloudWatch 日志执行的操作

以下代码示例演示了如何使用 AWS 软件开发工具包执行单个 CloudWatch 日志操作。这些摘录调用 CloudWatch Logs API，是必须在上下文中运行的大型程序的代码摘录。每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

以下示例仅包括最常用的操作。如需完整列表，请参阅 Amazon CloudWatch 日志 API 参考。

示例

- AssociateKmsKey与 AWS SDK 或 CLI 配合使用
- CancelExportTask与 AWS SDK 或 CLI 配合使用
- CreateExportTask与 AWS SDK 或 CLI 配合使用
- CreateLogGroup与 AWS SDK 或 CLI 配合使用
- CreateLogStream与 AWS SDK 或 CLI 配合使用
- DeleteLogGroup与 AWS SDK 或 CLI 配合使用
- DeleteSubscriptionFilter与 AWS SDK 或 CLI 配合使用
- DescribeExportTasks与 AWS SDK 或 CLI 配合使用
- DescribeLogGroups与 AWS SDK 或 CLI 配合使用
- DescribeSubscriptionFilters与 AWS SDK 或 CLI 配合使用
- GetQueryResults与 AWS SDK 或 CLI 配合使用
- PutSubscriptionFilter与 AWS SDK 或 CLI 配合使用
- StartLiveTail与 AWS SDK 或 CLI 配合使用
- StartQuery与 AWS SDK 或 CLI 配合使用

`AssociateKmsKey`与 AWS SDK 或 CLI 配合使用

下面的代码示例演示如何使用 `AssociateKmsKey`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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。

```
    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 的详细信息，请参阅 AWS SDK for .NET API 参考AssociateKmsKey中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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。

`CancelExportTask`与 AWS SDK 或 CLI 配合使用

下面的代码示例演示如何使用 `CancelExportTask`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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。

```
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 的详细信息，请参阅 AWS SDK for .NET API 参考CancelExportTask中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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。

CreateExportTask与 AWS SDK 或 CLI 配合使用

下面的代码示例演示如何使用 CreateExportTask。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to create an Export Task to export the contents of the Amazon
    /// CloudWatch Logs to the specified Amazon Simple Storage Service (Amazon
S3)
    /// bucket.
    /// </summary>
    public class CreateExportTask
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();
            string taskName = "export-task-example";
            string logGroupName = "cloudwatchlogs-example-loggroup";
            string destination = "doc-example-bucket";
            var fromTime = 1437584472382;
            var toTime = 1437584472833;

            var request = new CreateExportTaskRequest
            {
                From = fromTime,
                To = toTime,
                TaskName = taskName,
                LogGroupName = logGroupName,
```

```
                Destination = destination,
            };

            var response = await client.CreateExportTaskAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"The task, {taskName} with ID: " +
                                  $"{response.TaskId} has been created
successfully.");
            }
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考CreateExportTask中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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。

`CreateLogGroup`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateLogGroup`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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。

```
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 的详细信息，请参阅 AWS SDK for .NET API 参考CreateLogGroup中的。

CLI

AWS 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 的详细信息，请参阅AWS CLI 命令参考CreateLogGroup中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 的详细信息，请参阅 AWS SDK for JavaScript API 参考CreateLogGroup中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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。

`CreateLogStream`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateLogStream`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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。

```
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 的详细信息，请参阅 AWS SDK for .NET API 参考CreateLogStream中的。

CLI

AWS 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 的详细信息，请参阅AWS CLI 命令参考CreateLogStream中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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。

DeleteLogGroup与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteLogGroup。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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。

```
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 的详细信息，请参阅 AWS SDK for .NET API 参考DeleteLogGroup中的。

CLI

AWS 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 的详细信息，请参阅AWS CLI 命令参考DeleteLogGroup中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 的详细信息，请参阅 AWS SDK for JavaScript API 参考DeleteLogGroup中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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。

DeleteSubscriptionFilter与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteSubscriptionFilter。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 的详细信息，请参阅 AWS SDK for C++ API 参考DeleteSubscriptionFilter中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteSubscriptionFilter中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 的详细信息，请参阅 AWS SDK for JavaScript API 参考DeleteSubscriptionFilter中的。

适用于 JavaScript (v2) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 AWS SDK for JavaScript 开发人员指南。
- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DeleteSubscriptionFilter中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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。

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

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

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

- 有关 API 的详细信息，请参阅适用DeleteSubscriptionFilter于 K otlin 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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。

`DescribeExportTasks`与 AWS SDK 或 CLI 配合使用

下面的代码示例演示如何使用 `DescribeExportTasks`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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。

```
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 的详细信息，请参阅 AWS SDK for .NET API 参考DescribeExportTasks中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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。

DescribeLogGroups与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeLogGroups。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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。

```
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 的详细信息，请参阅 AWS SDK for .NET API 参考[DescribeLogGroups](#)中的。

CLI

AWS 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 的详细信息，请参阅AWS CLI 命令参考DescribeLogGroups中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 && page.logGroups.every((lg) => !!lg)) {
      logGroups.push(...page.logGroups);
    }
  }
```

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

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DescribeLogGroups中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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。

`DescribeSubscriptionFilters`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeSubscriptionFilters`。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 的详细信息，请参阅 AWS SDK for C++ API 参考DescribeSubscriptionFilters中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeSubscriptionFilters中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 的详细信息，请参阅 AWS SDK for JavaScript API 参考[DescribeSubscriptionFilters](#)中的。

适用于 JavaScript (v2) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 = {
  logGroupName: "GROUP_NAME",
  limit: 5,
};

cwl.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 AWS SDK for JavaScript 开发人员指南。
- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DescribeSubscriptionFilters中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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。

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient { region = "us-west-2" }.use { cwlClient ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name  ${filter.filterName} pattern
 ${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用DescribeSubscriptionFilters于 K otlin 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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。

GetQueryResults与 AWS SDK 或 CLI 配合使用

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

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 的详细信息，请参阅 AWS SDK for JavaScript API 参考GetQueryResults中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 的详细信息，请参阅适用GetQueryResults于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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。

PutSubscriptionFilter与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutSubscriptionFilter。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 的详细信息，请参阅 AWS SDK for C++ API 参考PutSubscriptionFilter中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 cwl = CloudWatchLogsClient.builder()
                .region(region)
                .build();

        putSubFilters(cwl, filter, pattern, logGroup, functionArn);
        cwl.close();
    }

    public static void putSubFilters(CloudWatchLogsClient cwl,
            String filter,
            String pattern,
            String logGroup,
            String functionArn) {

        try {
            PutSubscriptionFilterRequest request =
 PutSubscriptionFilterRequest.builder()
                    .filterName(filter)
```

```
                    .filterPattern(pattern)
                    .logGroupName(logGroup)
                    .destinationArn(functionArn)
                    .build();

            cwl.putSubscriptionFilter(request);
            System.out.printf(
                    "Successfully created CloudWatch logs subscription filter
 %s",
                    filter);

        } catch (CloudWatchLogsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutSubscriptionFilter中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 AWS SDK for JavaScript API 参考[PutSubscriptionFilter](#)中的。

适用于 JavaScript (v2) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 [AWS 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 AWS SDK for JavaScript 开发人员指南。
- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考PutSubscriptionFilter中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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。

`StartLiveTail`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `StartLiveTail`。

.NET

AWS SDK for .NET

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 的详细信息，请参阅 AWS SDK for .NET API 参考StartLiveTail中的。

Go

适用于 Go V2 的 SDK

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 occured 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 的详细信息，请参阅 AWS SDK for Go API 参考StartLiveTail中的。

Java

适用于 Java 2.x 的 SDK

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartLiveTail中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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 的详细信息，请参阅 AWS SDK for JavaScript API 参考StartLiveTail中的。

Kotlin

适用于 Kotlin 的 SDK

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 unsubcribe from the flow. This will:
                * 1). Close the stream
                * 2). Stop the Live Tail session
                */
                stream.takeWhile { System.currentTimeMillis() - startTime <
10000 }.collect { value ->
                    if (value is StartLiveTailResponseStream.SessionStart) {
                        println(value.asSessionStart())
                    } else if (value is
StartLiveTailResponseStream.SessionUpdate) {
                        for (e in value.asSessionUpdate().sessionResults!!) {
                            println(e)
                        }
                    } else {
                        throw IllegalArgumentException("Unknown event type")
                    }
                }
            } else {
                throw IllegalArgumentException("No response stream")
            }
        }
    } catch (e: Exception) {
        println("Exception occurred during StartLiveTail: $e")
        System.exit(1)
    }
```

- 有关 API 的详细信息，请参阅适用StartLiveTail于 K otlin 的AWS SDK API 参考。

Python

SDK for Python (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 的详细信息，请参阅适用StartLiveTail于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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。

`StartQuery`与 AWS SDK 或 CLI 配合使用

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

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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 的详细信息，请参阅 AWS SDK for JavaScript API 参考StartQuery中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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_groups,
                    startTime=start_time,
                    endTime=end_time,
                    queryString="fields @timestamp, @message | sort @timestamp
asc",
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(
```

```
self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_groups,
                startTime=start_time,
                endTime=end_time,
                queryString="fields @timestamp, @message | sort @timestamp asc",
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")
```

- 有关 API 的详细信息，请参阅适用StartQuery于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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 AWS SDK 的 CloudWatch 日志场景

以下代码示例向您展示了如何在 L CloudWatch ogs 中使用 AWS SDK 实现常见场景。这些场景向您展示了如何通过调用 L CloudWatch ogs 中的多个函数来完成特定任务。每个场景都包含一个指向的链接 GitHub，您可以在其中找到有关如何设置和运行代码的说明。

示例

- 使用 CloudWatch 日志运行大型查询

使用 CloudWatch 日志运行大型查询

以下代码示例展示了如何使用 CloudWatch 日志查询超过 10,000 条记录。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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(parseInt(process.env.QUERY_START_DATE)),
    new Date(parseInt(process.env.QUERY_END_DATE)),
  ],
});

await cloudWatchQuery.run();

console.log(
  `Queries finished in ${cloudWatchQuery.secondsElapsed} seconds.\nTotal logs
 found: ${cloudWatchQuery.results.length}`,
);
```

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]);
}

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

/**
 * Starts a query and waits for it to complete.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 */
async _query(dateRange, maxLogs) {
  try {
    const { queryId } = await this._startQuery(dateRange, maxLogs);
    const { results } = await this._waitUntilQueryDone(queryId);
    return results ?? [];
  } catch (err) {
```

```
      /**
       * This error is thrown when StartQuery returns an error indicating
       * that the query's start or end date occur before the log group was
       * created.
       */
      if (err instanceof DateOutOfBoundsError) {
        return [];
      } else {
        throw err;
      }
    }
  }

  // snippet-start:[javascript.v3.cloudwatch-logs.actions.StartQuery]
  /**
   * Wrapper for the StartQueryCommand. Uses a static query string
   * for consistency.
   * @param {[Date, Date]} dateRange
   * @param {number} maxLogs
   * @returns {Promise<{ queryId: string }>}
   */
  async _startQuery([startDate, endDate], maxLogs = 10000) {
    try {
      return await this.client.send(
        new StartQueryCommand({
          logGroupNames: this.logGroupNames,
          queryString: "fields @timestamp, @message | sort @timestamp asc",
          startTime: startDate.valueOf(),
          endTime: endDate.valueOf(),
          limit: maxLogs,
        }),
      );
    } catch (err) {
      /** @type {string} */
      const message = err.message;
      if (message.startsWith("Query's end date and time")) {
        // This error indicates that the query's start or end date occur
        // before the log group was created.
        throw new DateOutOfBoundsError(message);
      }

      throw err;
    }
  }
```

```
  // snippet-end:[javascript.v3.cloudwatch-logs.actions.StartQuery]

  /**
   * Call GetQueryResultsCommand until the query is done.
   * @param {string} queryId
   */
  _waitUntilQueryDone(queryId) {
    const getResults = async () => {
      const results = await this._getQueryResults(queryId);
      const queryDone = [
        "Complete",
        "Failed",
        "Cancelled",
        "Timeout",
        "Unknown",
      ].includes(results.status);

      return { queryDone, results };
    };

    return retry(
      { intervalInMs: 1000, maxRetries: 60, quiet: true },
      async () => {
        const { queryDone, results } = await getResults();
        if (!queryDone) {
          throw new Error("Query not done.");
        }

        return results;
      },
    );
  }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的以下主题。
  - GetQueryResults
  - StartQuery

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 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，用于管理超过 10,000 个结果的 CloudWatch 查询。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import logging
import os
import sys

import boto3
from botocore.config import Config

from cloudwatch_query import CloudWatchQuery
from date_utilities import DateUtilities

# Configure logging at the module level.
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(filename)s:%(lineno)d - %(message)s",
)


class CloudWatchLogsQueryRunner:
    def __init__(self):
        """
        Initializes the CloudWatchLogsQueryRunner class by setting up date
 utilities
        and creating a CloudWatch Logs client with retry configuration.
        """
        self.date_utilities = DateUtilities()
        self.cloudwatch_logs_client = self.create_cloudwatch_logs_client()

    def create_cloudwatch_logs_client(self):
        """
```

```
        Creates and returns a CloudWatch Logs client with a specified retry
configuration.

        :return: A CloudWatch Logs client instance.
        :rtype: boto3.client
        """
        try:
            return boto3.client("logs", config=Config(retries={"max_attempts":
10}))
        except Exception as e:
            logging.error(f"Failed to create CloudWatch Logs client: {e}")
            sys.exit(1)

    def fetch_environment_variables(self):
        """
        Fetches and validates required environment variables for query start and
end dates.

        :return: Tuple of query start date and end date as integers.
        :rtype: tuple
        :raises SystemExit: If required environment variables are missing or
invalid.
        """
        try:
            query_start_date = int(os.environ["QUERY_START_DATE"])
            query_end_date = int(os.environ["QUERY_END_DATE"])
        except KeyError:
            logging.error(
                "Both QUERY_START_DATE and QUERY_END_DATE environment variables
are required."
            )
            sys.exit(1)
        except ValueError as e:
            logging.error(f"Error parsing date environment variables: {e}")
            sys.exit(1)

        return query_start_date, query_end_date

    def convert_dates_to_iso8601(self, start_date, end_date):
        """
        Converts UNIX timestamp dates to ISO 8601 format using DateUtilities.

        :param start_date: The start date in UNIX timestamp.
        :type start_date: int
```

```
        :param end_date: The end date in UNIX timestamp.
        :type end_date: int
        :return: Start and end dates in ISO 8601 format.
        :rtype: tuple
        """
        start_date_iso8601 =
 self.date_utilities.convert_unix_timestamp_to_iso8601(
            start_date
        )
        end_date_iso8601 = self.date_utilities.convert_unix_timestamp_to_iso8601(
            end_date
        )
        return start_date_iso8601, end_date_iso8601

    def execute_query(
        self,
        start_date_iso8601,
        end_date_iso8601,
        log_group="/workflows/cloudwatch-logs/large-query",
    ):
        """
        Creates a CloudWatchQuery instance and executes the query with provided
 date range.

        :param start_date_iso8601: The start date in ISO 8601 format.
        :type start_date_iso8601: str
        :param end_date_iso8601: The end date in ISO 8601 format.
        :type end_date_iso8601: str
        :param log_group: Log group to search: "/workflows/cloudwatch-logs/large-
query"
        :type log_group: str
        """
        cloudwatch_query = CloudWatchQuery(
            [start_date_iso8601, end_date_iso8601],
        )
        cloudwatch_query.query_logs((start_date_iso8601, end_date_iso8601))
        logging.info("Query executed successfully.")
        logging.info(
            f"Queries completed in {cloudwatch_query.query_duration} seconds.
 Total logs found: {len(cloudwatch_query.query_results)}"
        )


def main():
```

```
    """
    Main function to start a recursive CloudWatch logs query.
    Fetches required environment variables, converts dates, and executes the
 query.
    """
    logging.info("Starting a recursive CloudWatch logs query...")
    runner = CloudWatchLogsQueryRunner()
    query_start_date, query_end_date = runner.fetch_environment_variables()
    start_date_iso8601 = DateUtilities.convert_unix_timestamp_to_iso8601(
        query_start_date
    )
    end_date_iso8601 =
 DateUtilities.convert_unix_timestamp_to_iso8601(query_end_date)
    runner.execute_query(start_date_iso8601, end_date_iso8601)


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

此模块处理超过 10,000 个结果的 CloudWatch 查询。

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

from date_utilities import DateUtilities


class DateOutOfBoundsError(Exception):
    """Exception raised when the date range for a query is out of bounds."""

    pass


class CloudWatchQuery:
    """
    A class to query AWS CloudWatch logs within a specified date range.
```

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

    def __init__(self, date_range):
        self.lock = threading.Lock()
        self.log_groups = "/workflows/cloudwatch-logs/large-query"
        self.query_results = []
        self.date_range = date_range
        self.query_duration = None
        self.datetime_format = "%Y-%m-%d %H:%M:%S.%f"
        self.date_utilities = DateUtilities()
        self.limit = 10000

    def query_logs(self, date_range):
        """
        Executes a CloudWatch logs query for a specified date range and
calculates the execution time of the query.

        :return: A batch of logs retrieved from the CloudWatch logs query.
        :rtype: list
        """
        start_time = datetime.now()

        start_date, end_date = self.date_utilities.normalize_date_range_format(
            date_range, from_format="unix_timestamp", to_format="datetime"
        )

        logging.info(
            f"Original query:"
            f"\n       START:    {start_date}"
            f"\n       END:      {end_date}"
        )
        self.recursive_query((start_date, end_date))
        end_time = datetime.now()
        self.query_duration = (end_time - start_time).total_seconds()

    def recursive_query(self, date_range):
        """
        Processes logs within a given date range, fetching batches of logs
recursively if necessary.
```

```
        :param date_range: The date range to fetch logs for, specified as a tuple
(start_timestamp, end_timestamp).
        :type date_range: tuple
        :return: None if the recursive fetching is continued or stops when the
final batch of logs is processed.
                 Although it doesn't explicitly return the query results, this
method accumulates all fetched logs
                 in the `self.query_results` attribute.
        :rtype: None
        """
        batch_of_logs = self.perform_query(date_range)
        # Add the batch to the accumulated logs
        with self.lock:
            self.query_results.extend(batch_of_logs)
        if len(batch_of_logs) == self.limit:
            logging.info(f"Fetched {self.limit}, checking for more...")
            most_recent_log = self.find_most_recent_log(batch_of_logs)
            most_recent_log_timestamp = next(
                item["value"]
                for item in most_recent_log
                if item["field"] == "@timestamp"
            )
            new_range = (most_recent_log_timestamp, date_range[1])
            midpoint = self.date_utilities.find_middle_time(new_range)

            first_half_thread = threading.Thread(
                target=self.recursive_query,
                args=((most_recent_log_timestamp, midpoint),),
            )
            second_half_thread = threading.Thread(
                target=self.recursive_query, args=((midpoint, date_range[1]),)
            )

            first_half_thread.start()
            second_half_thread.start()

            first_half_thread.join()
            second_half_thread.join()

    def find_most_recent_log(self, logs):
        """
        Search a list of log items and return most recent log entry.
        :param logs: A list of logs to analyze.
        :return: log
```

```
        :type :return List containing log item details
        """
        most_recent_log = None
        most_recent_date = "1970-01-01 00:00:00.000"

        for log in logs:
            for item in log:
                if item["field"] == "@timestamp":
                    logging.debug(f"Compared: {item['value']} to
{most_recent_date}")
                    if (
                        self.date_utilities.compare_dates(
                            item["value"], most_recent_date
                        )
                        == item["value"]
                    ):
                        logging.debug(f"New most recent: {item['value']}")
                        most_recent_date = item["value"]
                        most_recent_log = log
        logging.info(f"Most recent log date of batch: {most_recent_date}")
        return most_recent_log

    # snippet-start:[python.example_code.cloudwatch_logs.start_query]
    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
```

```
                response = client.start_query(
                    logGroupName=self.log_groups,
                    startTime=start_time,
                    endTime=end_time,
                    queryString="fields @timestamp, @message | sort @timestamp
asc",
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(
```

```
self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_groups,
                startTime=start_time,
                endTime=end_time,
                queryString="fields @timestamp, @message | sort @timestamp asc",
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")

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

    # snippet-start:[python.example_code.cloudwatch_logs.get_query_results]
    def _wait_for_query_results(self, client, query_id):
        """
        Waits for the query to complete and retrieves the results.

        :param query_id: The ID of the initiated query.
        :type query_id: str
        :return: A list containing the results of the query.
        :rtype: list
        """
        while True:
            time.sleep(1)
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",
                "Cancelled",
                "Timeout",
                "Unknown",
            ]:
                return results.get("results", [])

    # snippet-end:[python.example_code.cloudwatch_logs.get_query_results]
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的以下主题。
  - GetQueryResults

- StartQuery

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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 AWS SDK 的 CloudWatch 日志的跨服务示例

以下示例应用程序使用 AWS SDK 将 CloudWatch 日志与其他 AWS 服务应用程序组合在一起。每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何设置和运行应用程序的说明。

示例

- 使用计划的事件调用 Lambda 函数

使用计划的事件调用 Lambda 函数

以下代码示例说明如何创建由 Amazon EventBridge 计划事件调用的 AWS Lambda 函数。

Python

SDK for Python (Boto3)

此示例说明如何将 AWS Lambda 函数注册为计划的 Amazon EventBridge 事件的目标。Lambda 处理程序将友好的消息和完整的事件数据写入 Amazon CloudWatch 日志，以供日后检索。

- 部署 Lambda 函数。
- 创建 EventBridge 计划事件并将 Lambda 函数设为目标。
- 授予允许 EventBridge 调用 Lambda 函数的权限。
- 打印来自 CloudWatch Logs 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。

最好在上查看此示例 GitHub。有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- CloudWatch 日志
- EventBridge

- Lambda

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅在 AWS SDK 中使用 CloudWatch 日志。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 日志中的安全

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将此描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用的合规计划 WorkSpaces，请参阅按合规计划划分的范围内的AWSAWS 服务按合规计划。
- 云端安全-您的责任由您使用的 AWS 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、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用 Amazon Logs 时如何应用分担责任 CloudWatch 模型。它向您展示了如何配置 Amazon CloudWatch Logs 以满足您的安全和合规目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 CloudWatch 日志资源。

内容

- Amazon CloudWatch 日志中的数据保护
- Amazon CloudWatch 日志的身份和访问管理
- Amazon CloudWatch 日志的合规性验证
- Amazon CloudWatch Logs 中的恢复能力
- Amazon CloudWatch 日志中的基础设施安全
- 在接口 VPC 终端节点上使用 CloudWatch 日志

Amazon CloudWatch 日志中的数据保护

> **Note**
>
> 除了中有关常规数据保护的以下信息外 AWS， CloudWatch 日志还允许您通过屏蔽日志事件中的敏感数据来保护这些数据。有关更多信息，请参阅 通过屏蔽帮助保护敏感的日志数据。

责任 AWS 共担模式分适用于 Amazon CloudWatch Logs 中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础设施上的内容的控制。您还负责您所使用的 AWS 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。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的 AWS 责任共担模式和 GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。我们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 与资源通信。 AWS 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 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。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-2 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 140-2 版》。

我们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 AWS 服务 使用控制台、API 或 AWS SDK 处理 CloudWatch 日志或其他内容时。 AWS CLI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 日志使用加密保护静态数据。所有日志组均加密。默认情况下， CloudWatch 日志服务管理服务器端加密密钥。

如果要管理用于加密和解密日志的密钥，请使用密钥。 AWS KMS 有关更多信息，请参阅 使用加密日志中的 CloudWatch 日志数据 AWS Key Management Service。

传输中加密

CloudWatch 日志对 end-to-end 传输中的数据进行加密。 CloudWatch 日志服务管理服务器端的加密密钥。

Amazon CloudWatch 日志的身份和访问管理

访问 Amazon CloudWatch Logs 需要凭证 AWS 才能对您的请求进行身份验证。这些凭证必须具有访问 AWS 资源的权限，例如检索有关您的云资源的 CloudWatch 日志数据。以下各节详细介绍了如何使用 AWS Identity and Access Management (IAM) 和 CloudWatch 日志，通过控制谁可以访问资源来保护您的资源：

- 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：

- 中的用户和群组 AWS IAM Identity Center：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。
- 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》中为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

您可以拥有有效的凭证来验证您的请求，但是除非您拥有权限，否则您无法创建或访问 CloudWatch 日志资源。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。

以下各节介绍如何管理 CloudWatch 日志的权限。我们建议您先阅读概述。

- 管理您的 L CloudWatch ogs 资源的访问权限概述
- 对日志使用基于身份的策略（IAM 策略） CloudWatch
- CloudWatch 日志权限参考

管理您的 L CloudWatch ogs 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：

- 中的用户和群组 AWS IAM Identity Center：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。
- 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》中为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 记录资源和操作
- 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 记录资源和操作

在 CloudWatch 日志中，主要资源是日志组、日志流和目标。 CloudWatch 日志不支持子资源（用于主资源的其他资源）。

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。第二个命令结尾为，是 `describe-log-groups` CLI 命令和 DescribeLogGroupsAPI 返回的内容。`:*`

资源类型	ARN 格式
	arn:aws:logs:*region*:*account-id* :log-group:*log_group_name* arn:aws:logs:*region*:*account-id* :log-group:*log_group_name* :* 在以下情况下，使用第一个版本，不带尾随字:*符： • 在许多 CloudWatch Logs API 的`logGroupIdentifier` 输入字段中。 • 在标记 `resourceArn` API 的字段中 • 在 IAM 策略中，当为TagResourceUntagResource、和指定权限时ListTagsForResource。 在 IAM 策略中为所有其他 API 操作指定权限时:*，使用带有尾随的第二个版本来引用 ARN。
日志流	*arn: aws: logs: ##### ID#####log_group_name_log-stream# log-stream-name*
目标位置	arn:aws:logs:*region*:*account-id* :destination:*destination_name*

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 的信息，请参阅中的 Amazon 资源名称 (ARN)。Amazon Web Services 一般参考有关涵盖CloudWatch 日志的策略的示例，请参阅对日志使用基于身份的策略（IAM 策略） CloudWatch 。

CloudWatch 日志提供了一组使用 CloudWatch 日志资源的操作。有关可用操作的列表，请参阅 CloudWatch 日志权限参考。

了解资源所有权

该 AWS 账户拥有在账户中创建的资源，无论谁创建了这些资源。具体而言，资源所有者是对 AWS 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）的账户。以下示例说明了它的工作原理：

- 如果您使用账户的根账户凭证创建日志组，则您的 AWS 账户就是 CloudWatch 日志资源的所有者。AWS
- 如果您在 AWS 账户中创建用户并向该用户授予创建 CloudWatch 日志资源的权限，则该用户可以创建 CloudWatch 日志资源。但是，该用户所属的您的 AWS 账户拥有 CloudWatch 日志资源。
- 如果您在 AWS 账户中创建具有创建 CloudWatch 日志资源的权限的 IAM 角色，则任何能够担任该角色的人都可以创建 CloudWatch 日志资源。该角色所属的您的 AWS 账户拥有 CloudWatch 日志资源。

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**
>
> 本节讨论在 CloudWatch 日志上下文中使用 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 策略语法和说明的信息，请参阅 IAM 用户指南中的 IAM 策略参考。

附加到 IAM 身份的策略称为基于身份的策略（IAM 策略），附加到资源的策略称为基于资源的策略。CloudWatch 日志支持基于身份的策略和基于资源的目标策略，这些策略用于启用跨账户订阅。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 是一项 CloudWatch 功能，它使您能够分析来自日志组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权限时，该用户可以创建一个规则来评估日志中的 CloudWatch 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 日志支持针对目标的基于资源的策略，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 步骤 1：创建目标。可以使用 PutDestinationAPI 创建目的地，也可以使用 AP PutDestinationPolicyI 向目的地添加资源策略。以下示例允许 AWS 账户编号为 111122223333 的另一个账户将其日志组订阅到目标。arn:aws:logs:us-east-1:123456789012:destination:testDestination

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111122223333"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-east-1:123456789012:destination:testDestination"
    }
  ]
}
```

指定策略元素：操作、效果和主体

对于每个 CloudWatch Logs 资源，该服务都定义了一组 API 操作。为了授予这些 API 操作的权限，CloudWatch 日志定义了一组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 记录资源和操作 和 CloudWatch 日志权限参考。

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 记录资源和操作。
- 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，您可以指定要获得权限的用户、账户、服务或其他实体（仅适用于基于资源的策略）。 CloudWatch 日志支持基于资源的目标策略。

有关 IAM 策略语法和介绍的更多信息，请参阅《IAM 用户指南》中的 AWS IAM 策略参考。

有关显示所有 L CloudWatch ogs API 操作及其适用的资源的表格，请参阅CloudWatch 日志权限参考。

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。有关每项 AWS 服务支持的上下文密钥列表以及 AWS全局条件上下文密钥列表，请参阅AWS 服务的操作、资源和条件键以及AWS 全局条件上下文密钥。

> **Note**
>
> 您可以使用标签来控制对 CloudWatch 日志资源的访问权限，包括日志组和目标。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。

对日志使用基于身份的策略（IAM 策略） CloudWatch

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**
>
> 我们建议您先阅读介绍性主题，这些主题解释了管理 CloudWatch 日志资源访问权限的基本概念和选项。有关更多信息，请参阅 管理您的 L CloudWatch ogs 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
- AWS CloudWatch 日志的托管（预定义）策略
- 客户管理型策略示例

下面是权限策略的示例：

```
{
  "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 日志操作的列表，请参阅CloudWatch 日志权限参考。

使用 CloudWatch 控制台所需的权限

要让用户在 CloudWatch 控制台中使用 CloudWatch 日志，该用户必须拥有一组允许用户描述其 AWS 账户中的其他 AWS 资源的最低权限。要在 CloudWatch 控制台中使用 CloudWatch 日志，您必须拥有以下服务的权限：

- CloudWatch
- CloudWatch 日志
- OpenSearch 服务
- 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。为确保这些用户仍然可以使用 CloudWatch 控制台，还要将`CloudWatchReadOnlyAccess`托管策略附加到该用户，如中所述AWS CloudWatch 日志的托管（预定义）策略。

对于仅调用 AWS CLI 或 CloudWatch 日志 API 的用户，您无需为其设置最低控制台权限。

对于不使用 CloudWatch 控制台管理日志订阅的用户，使用控制台所需的全部权限为：

- 云观察：GetMetricData
- 云观察：ListMetrics
- 日志：CancelExportTask
- 日志：CreateExportTask
- 日志：CreateLogGroup
- 日志：CreateLogStream
- 日志：DeleteLogGroup
- 日志：DeleteLogStream
- 日志：DeleteMetricFilter

- 日志：DeleteQueryDefinition
- 日志：DeleteRetentionPolicy
- 日志：DeleteSubscriptionFilter
- 日志：DescribeExportTasks
- 日志：DescribeLogGroups
- 日志：DescribeLogStreams
- 日志：DescribeMetricFilters
- 日志：DescribeQueryDefinitions
- 日志：DescribeQueries
- 日志：DescribeSubscriptionFilters
- 日志：FilterLogEvents
- 日志：GetLogEvents
- 日志：GetLogGroupFields
- 日志：GetLogRecord
- 日志：GetQueryResults
- 日志：PutMetricFilter
- 日志：PutQueryDefinition
- 日志：PutRetentionPolicy
- 日志：StartQuery
- 日志：StopQuery
- 日志：PutSubscriptionFilter
- 日志：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：

- 是：DescribeElasticsearchDomain
- 是：ListDomainNames
- 我是：AttachRolePolicy
- 我是：CreateRole
- 我是：GetPolicy
- 我是：GetPolicyVersion

- 我是：GetRole
- 我是：ListAttachedRolePolicies
- 我是：ListRoles
- 运动学:DescribeStreams
- 运动学:ListStreams
- lambda: AddPermission
- lambda: CreateFunction
- lambda: GetFunctionConfiguration
- lambda: ListAliases
- lambda: ListFunctions
- lambda: ListVersionsByFunction
- lambda: RemovePermission
- s3：ListBuckets

AWS CloudWatch 日志的托管（预定义）策略

AWS 通过提供由创建和管理的独立 IAM 策略来解决许多常见用例 AWS。托管式策略可授予常用案例的必要权限，因此，您可以免去调查都需要哪些权限的工作。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略。

以下 AWS 托管策略特定于 CloudWatch 日志，您可以将其附加到账户中的用户和角色：

- CloudWatchLogsFullAccess— 授予对 CloudWatch 日志的完全访问权限。
- CloudWatchLogsReadOnlyAccess— 授予对CloudWatch 日志的只读访问权限。

CloudWatchLogsFullAccess

该CloudWatchLogsFullAccess策略授予对 CloudWatch 日志的完全访问权限。该策略包含`cloudwatch:GenerateQuery`权限，因此拥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。内容如下：

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

```
                "logs:*",
                "cloudwatch:GenerateQuery"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

CloudWatchLogsReadOnlyAccess

该CloudWatchLogsReadOnlyAccess策略授予对 CloudWatch 日志的只读访问权限。它包括`cloudwatch:GenerateQuery`权限，因此拥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。内容如下：

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

CloudWatchLogsCrossAccountSharingConfiguration

该CloudWatchLogsCrossAccountSharingConfiguration策略授予创建、管理和查看用于在账户之间共享 CloudWatch 日志资源的 Observability Access Manager 链接的权限。有关更多信息，请参阅CloudWatch 跨账户可观察性。

内容如下：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:Link",
                "oam:ListLinks"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:DeleteLink",
                "oam:GetLink",
                "oam:TagResource"
            ],
            "Resource": "arn:aws:oam:*:*:link/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink"
            ],
            "Resource": [
                "arn:aws:oam:*:*:link/*",
                "arn:aws:oam:*:*:sink/*"
            ]
        }
    ]
}
```

CloudWatch 记录 AWS 托管策略的更新

查看自该服务开始跟踪 CloudWatch 日志 AWS 托管策略更改以来这些更新的详细信息。要获得有关此页面更改的自动提醒，请订阅 “ CloudWatch 日志文档历史记录” 页面上的 RSS feed。

更改	描述	日期
CloudWatchLogsFullAccess – 对现有策略的更新。	CloudWatch 日志已向添加权限CloudWatchLogsFullAccess。 此`cloudwatch:GenerateQuery` 权限已添加，因此拥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 27 日
CloudWatchLogsReadOnlyAccess – 对现有策略的更新。	CloudWatch 已向。添加了权限CloudWatchLogsReadOnlyAccess。 此`cloudwatch:GenerateQuery` 权限已添加，因此拥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 27 日
CloudWatchLogsReadOnlyAccess – 更新了现有策略	CloudWatch 记录已添加的权限CloudWatchLogsReadOnlyAccess。 添加了`logs:StartLiveTail` 和`logs:StopLiveTail` 权限，以便拥有此策略的用户可以使用控制台启动和停止 L CloudWatch ogs 实时尾部会话。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
		2022 年 11 月 27 日

更改	描述	日期
CloudWatchLogsCrossAccountSharingConfiguration：新策略	CloudWatch Logs 添加了一项新策略，使您能够管理共享 CloudWatch 日志组的 CloudWatch 跨账户可观察性链接。 如需了解更多信息，请参阅CloudWatch 跨账户可观察性	
CloudWatchLogsReadOnlyAccess – 更新了现有策略	CloudWatch 记录已添加的权限CloudWatchLogsReadOnlyAccess。 添加了`oam:ListSinks` 和`oam:ListAttachedLinks` 权限，以便拥有此策略的用户可以使用控制台在 CloudWatch 跨账户可观察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

您可以创建自己的自定义 IAM 策略以授予 CloudWatch 日志操作和资源的权限。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。

在本节中，您可以找到授予各种CloudWatch 日志操作权限的用户策略示例。这些策略在您使用 CloudWatch 日志 API、 AWS 软件开发工具包或时起 AWS CLI作用。

示例

- 示例 1：允许完全访问 CloudWatch 日志
- 示例 2：允许对 CloudWatch 日志进行只读访问
- 示例 3：允许访问一个日志组

示例 1：允许完全访问 CloudWatch 日志

以下策略允许用户访问所有 CloudWatch 日志操作。

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

示例 2：允许对 CloudWatch 日志进行只读访问

AWS 提供了允许对 CloudWatch 日志数据进行只读访问的CloudWatchLogsReadOnlyAccess策略。该策略包含以下权限。

```
{
    "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。

```
{
   "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 日志中标记 CloudWatch 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 Team 标签键值为 Green 的日志组的访问权限。

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

```
            "Action": [
                "logs:*"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/Team": "Green"
                }
            }
        }
    ]
}
```

StopQuery和 StopLiveTailAPI 操作不与传统意义上的 AWS 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，它们仅对给定的实时跟踪会话或给定的 L CloudWatch ogs Insights 查询进行操作，这些查询未归类为资源。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。

```
{
    "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 日志权限参考

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。该表列出了每 CloudWatch 个 Logs API 操作以及您可以为其授予执行该操作的权限的相应操作。请在策略的 `Action` 字段中指定这些操作。对于该`Resource`字段，您可以指定日志组或日志流的 ARN，也可以指定*代表所有 CloudWatch 日志资源。

您可以在 CloudWatch 日志策略中使用 AWS-wide 条件键来表达条件。有关 AWS范围密钥的完整列表，请参阅 IAM 用户指南中的AWS 全局和 IAM 条件上下文密钥。

> **Note**
>
> 要指定操作，请在 API 操作名称之前使用 `logs:` 前缀。例如：`logs:CreateLogGrouplogs:CreateLogStream`、或`logs:*`（适用于所有 CloudWatch 日志操作）。

CloudWatch 记录 API 操作和操作所需的权限

CloudWatch 记录 API 的操作	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。
CreateExportTask	`logs:CreateExportTask` 将数据从日志组导出到 Amazon S3 存储桶所需。
CreateLogGroup	`logs:CreateLogGroup` 需要创建新的日志组。
CreateLogStream	`logs:CreateLogStream` 需要在日志组中创建新的日志流。
DeleteDestination	`logs:DeleteDestination` 需要删除日志目标并对其禁用所有订阅筛选器。
DeleteLogGroup	`logs:DeleteLogGroup` 需要删除日志组和所有关联的存档日志事件。
DeleteLogStream	`logs:DeleteLogStream` 需要删除日志流和所有关联的存档日志事件。

CloudWatch 记录 API 的操作	所需权限（API 操作）
DeleteMetricFilter	`logs:DeleteMetricFilter` 需要删除与日志组关联的指标筛选器。
DeleteQueryDefinition	`logs:DeleteQueryDefinition` 需要在 Logs Insight CloudWatch s 中删除已保存的查询定义。
DeleteResourcePolicy	`logs:DeleteResourcePolicy` 需要删除 CloudWatch 日志资源策略。
DeleteRetentionPolicy	`logs:DeleteRetentionPolicy` 需要删除日志组的保留策略。
DeleteSubscriptionFilter	`logs:DeleteSubscriptionFilter` 需要删除与日志组关联的订阅筛选器。
DescribeDestinations	`logs:DescribeDestinations` 需要查看与账户关联的所有目标。
DescribeExportTasks	`logs:DescribeExportTasks` 需要查看与账户关联的所有导出任务。
DescribeLogGroups	`logs:DescribeLogGroups` 需要查看与账户关联的所有日志组。
DescribeLogStreams	`logs:DescribeLogStreams` 需要查看与日志组关联的所有日志流。
DescribeMetricFilters	`logs:DescribeMetricFilters` 需要查看与日志组关联的所有指标。

CloudWatch 记录 API 的操作	所需权限（API 操作）
DescribeQueryDefinitions	`logs:DescribeQueryDefinitions` 需要在 Logs Insights 中 CloudWatch 查看已保存的查询定义列表。
DescribeQueries	`logs:DescribeQueries` 需要查看已计划、正在执行或最近执行的 L CloudWatch ogs Insights 查询列表。
DescribeResourcePolicies	`logs:DescribeResourcePolicies` 查看 CloudWatch 日志资源策略列表所必需的。
DescribeSubscriptionFilters	`logs:DescribeSubscriptionFilters` 需要查看与日志组关联的所有订阅筛选器。
FilterLogEvents	`logs:FilterLogEvents` 需要按照日志组筛选器模式对日志事件进行排序。
GetLogEvents	`logs:GetLogEvents` 需要从日志流中检索日志事件。
GetLogGroupFields	`logs:GetLogGroupFields` 需要检索日志组中日志事件内包含的字段列表。
GetLogRecord	`logs:GetLogRecord` 需要从单个日志事件中检索详细信息。
GetQueryResults	`logs:GetQueryResults` 检索 L CloudWatch ogs Insights 查询结果所必需的。

CloudWatch 记录 API 的操作	所需权限（API 操作）
ListTagsLogGroup	`logs:ListTagsLogGroup` 需要列出与日志组关联的标签。
PutDestination	`logs:PutDestination` 创建或更新目标日志流（例如，Kinesis 流）所需。
PutDestinationPolicy	`logs:PutDestinationPolicy` 需要创建或更新与现有日志目标关联的访问策略。
PutLogEvents	`logs:PutLogEvents` 需要将一批日志事件上传到日志流。
PutMetricFilter	`logs:PutMetricFilter` 需要创建或更新指标筛选器并将其与日志组关联。
PutQueryDefinition	`logs:PutQueryDefinition` 需要在“ CloudWatch 日志见解”中保存查询。
PutResourcePolicy	`logs:PutResourcePolicy` 创建 CloudWatch 日志资源策略所必需的。
PutRetentionPolicy	`logs:PutRetentionPolicy` 需要设置日志组中的日志事件的保存（保留）天数。

CloudWatch 记录 API 的操作	所需权限（API 操作）
PutSubscriptionFilter	`logs:PutSubscriptionFilter` 需要创建或更新订阅筛选器并将其与日志组关联。
StartQuery	`logs:StartQuery` 启动 CloudWatch 日志见解查询所必需的。
StopQuery	`logs:StopQuery` 需要停止正在进行的 CloudWatch Logs Insights 查询。
TagLogGroup	`logs:TagLogGroup` 需要添加或更新日志组标签。
TestMetricFilter	`logs:TestMetricFilter` 需要针对日志事件消息采样测试筛选器模式。

为 CloudWatch 日志使用服务相关角色

Amaz CloudWatch on Logs 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种独特的 IAM 角色，直接链接到 CloudWatch 日志。服务相关角色由 CloudWatch Logs 预定义，包括该服务代表您调用其他 AWS 服务所需的所有权限。

服务相关角色可以提高 CloudWatch 日志设置的效率，因为您无需手动添加必要的权限。 CloudWatch 日志定义了其服务相关角色的权限，除非另有定义，否则只有 CloudWatch 日志可以担任这些角色。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AWS 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。

日志的服务相关角色权限 CloudWatch

CloudWatch 日志使用名AWSServiceRoleForLogDelivery为的服务相关角色。CloudWatch 日志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 启用来自 AWS 服务的日志记录。

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：`firehose:PutRecord`以及`firehose:PutRecordBatch`所有带有`LogDeliveryEnabled`键值为的标签的 Firehose 直播中。`True`当您创建订阅以将日志传送到 Firehose 时，此标签会自动附加到 Firehose 直播中。

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 日志创建服务相关角色 CloudWatch

您无需手动创建服务相关角色。当您在 AWS Management Console、或 AWS API 中将日志设置为直接发送到 Firehose 流时， CloudWatch 日志会为您创建服务相关角色。 AWS CLI

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 日志会再次为您创建服务相关角色。

编辑日志的服务相关角色 CloudWatch

CloudWatch 日志不允许您在创建服务相关角色或任何其他服务相关角色后对其进行编辑AWSServiceRoleForLogDelivery。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。

删除日志的服务相关角色 CloudWatch

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**
>
> 如果您尝试删除资源时， CloudWatch 日志服务正在使用该角色，则删除可能会失败。如果发生这种情况，请等待几分钟后重试。

删除AWSServiceRoleForLogDelivery服务相关角色使用的 CloudWatch 日志资源

- 停止直接向 Firehose 直播发送日志。

使用 IAM 手动删除服务相关角色

使用 IAM 控制台 AWS CLI、或 AWS API 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 日志服务相关角色支持的区域

CloudWatch Logs 支持在提供服务的所有 AWS 区域中使用服务相关角色。有关更多信息，请参阅CloudWatch 记录区域和终端节点。

Amazon CloudWatch 日志的合规性验证

作为多项合规计划的一部分，第三方审计师评估 Ama CloudWatch zon Logs 的安全 AWS 性和合规性。其中包括 SOC、PCI、FedRAMP、HIPAA 及其他。

有关特定合规计划范围内的 AWS 服务列表，请参阅按合规计划划分的范围内的AWSAWS 服务按合规计划。有关常规信息，请参阅 AWS 合规性计划 。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 Amazon CloudWatch Logs 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全性与合规性快速入门指南 - 这些部署指南讨论了架构注意事项，并提供了在 AWS上部署基于安全性和合规性的基准环境的步骤。
- 在 A@@ mazon Web Services 上构建 HIPAA 安全与合规性 — 本白皮书描述了各公司如何使用 AWS 来创建符合 HIPAA 标准的应用程序。
- AWS 合规资源AWS — 此工作簿和指南集可能适用于您所在的行业和所在地区。

- 使用AWS Config 开发人员指南中的规则评估资源 — AWS Config; 评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 此 AWS 服务可全面了解您的安全状态 AWS ，帮助您检查是否符合安全行业标准和最佳实践。

Amazon CloudWatch Logs 中的恢复能力

AWS全球基础设施围绕AWS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、容错性和可扩展性。

有关AWS区域和可用区的更多信息，请参阅AWS全球基础设施。

Amazon CloudWatch 日志中的基础设施安全

作为一项托管服务，Amaz CloudWatch on Logs 受 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用通过网络访问 CloudWatch 日志。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。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

在接口 VPC 终端节点上使用 CloudWatch 日志

如果您使用亚马逊虚拟私有云（Amazon VPC）托管 AWS 资源，则可以在您的 VPC 和 CloudWatch 日志之间建立私有连接。您可以使用此连接将日志发送到日 CloudWatch 志，而无需通过互联网发送日志。

Amazon VPC 是一项 AWS 服务，可用于在您定义的虚拟网络中启动 AWS 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。要将您的 VPC 连接到 CloudWatch

日志，您需要为 CloudWatch 日志定义接口 VPC 终端节点。这种类型的端点使您能够将 VPC 连接到 AWS 服务。该端点无需互联网网关、网络地址转换 (NAT) 实例或 VPN 连接即可提供与 Lo CloudWatch gs 的可靠、可扩展的连接。有关更多信息，请参阅 Amazon VPC 用户指南中的什么是 Amazon VPC。

Interface VPC 终端节点由 AWS PrivateLink一种 AWS 技术提供支持，该技术使用带有私有 IP 地址的弹性网络接口实现 AWS 服务之间的私密通信。有关更多信息，请参阅新 AWS PrivateLink 增 AWS 服务。

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 日志目前支持所有 AWS 区域（包括区域）的 VPC 终端节点。 AWS GovCloud (US)

为 CloudWatch 日志创建 VPC 终端节点

要开始在您的 VPC 中使用 CloudWatch 日志，请为 CloudWatch 日志创建一个接口 VPC 终端节点。要选择的服务是 com.amazonaws.***Region***.logs。您无需更改 “ CloudWatch 日志” 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终端节点。

测试您的 VPC 和 CloudWatch 日志之间的连接

创建端点后，您可以测试连接。

测试您的 VPC 和您的 CloudWatch 日志终端节点之间的连接

1. 连接到位于您的 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 AWS 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。

控制对您的 CloudWatch 日志 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。

以下是 CloudWatch 日志端点策略的示例。此策略允许通过 VPC 连接到 CloudWatch 日志的用户创建日志流并将日志发送到 CloudWatch 日志，并阻止他们执行其他 CloudWatch 日志操作。

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

修改 CloudWatch 日志的 VPC 终端节点策略

1. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
2. 在导航窗格中，选择端点。

3. 如果您尚未为 CloudWatch 日志创建终端节点，请选择创建终端节点。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 日志支持可以限制对`aws:SourceVpc`特定 VPC 或特定 VPC 终端节点的访问权限的`aws:SourceVpce`上下文密钥。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。

记录 CloudWatch 日志 API 和控制台操作在 AWS CloudTrail

Amazon CloudWatch Logs 与 AWS CloudTrail一项服务集成，可在 CloudWatch 日志中记录用户、角色或 AWS 服务所采取的操作。 CloudTrail 捕获您的账户或代表您的 AWS 账户进行的 API 调用。捕获的调用包括来自 CloudWatch 控制台的调用和对 CloudWatch 日志 API 操作的代码调用。如果您创建跟踪，则可以将 CloudTrail事件持续传输到 Amazon S3 存储桶，包括 CloudWatch 日志事件。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的事件历史记录中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向 L CloudWatch ogs 发出的请求、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，包括如何配置和启用它，请参阅AWS CloudTrail 用户指南。

主题

- CloudWatch 将信息记录到 CloudTrail
- 查询生成信息 CloudTrail
- 了解日志文件条目

CloudWatch 将信息记录到 CloudTrail

CloudTrail 在您创建 AWS 账户时已在您的账户上启用。当 CloudWatch 日志中出现支持的事件活动时，该活动将与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在自己的 AWS 账户中查看、搜索和下载最近发生的事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录您的 AWS 账户中的事件，包括 CloudWatch 日志事件，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，当您在控制台中创建跟踪时，该跟踪将应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅下列内容：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和从多个账户接收 CloudTrail日志文件

CloudWatch 日志支持将以下操作作为事件记录在 CloudTrail 日志文件中：

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

只有这些 CloudWatch 日志 API 操作 CloudTrail 的请求元素才会被登录：

- DescribeDestinations
- DescribeExportTasks
- DescribeLogGroups
- DescribeLogStreams
- DescribeMetricFilters
- DescribeQueries
- DescribeResourcePolicies
- DescribeSubscriptionFilters
- FilterLogEvents

- GetLogEvents
- GetLogGroupFields
- GetLogRecord
- GetQueryResults

每个事件或日记账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 AWS 服务发出。

有关更多信息，请参阅CloudTrail 用户身份元素。

查询生成信息 CloudTrail

CloudTrail 还支持记录查询生成器控制台事件。 CloudWatch 日志见解和 CloudWatch 指标见解目前支持查询生成器。在这些 CloudTrail 事件中，`eventSource`是`monitoring.amazonaws.com`。

以下示例显示了一个演示 Logs Insights 中GenerateQuery操作的 CloudTrail CloudWatch 日志条目。

```
{
    "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，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序出现。

以下日志文件条目显示用户调用了 “日 CloudWatch 志 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"
}
```

CloudWatch 日志代理参考

> **Important**
>
> 此参考适用于较早已弃用的 CloudWatch Logs 代理。如果您使用实例元数据服务版本 2 (imdsv2)，则必须使用新的统一代理。 CloudWatch 即使您没有使用 imdsv2，我们也强烈建议您使用较新的统一 CloudWatch 代理而不是较旧的日志代理。有关更新的统一代理的更多信息，请参阅使用代理从 Amazon EC2 实例和本地服务器收集指标和日志。 CloudWatch 有关从较旧的 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 日志代理提供了一种自动将日志数据从 Amazon EC2 实例发送到 CloudWatch 日志的方法。该代理包括以下组件：

- 的插件，可 AWS CLI 将日志数据推送到 CloudWatch 日志。
- 一个脚本（守护程序），用于启动将数据推送到 Logs 的 CloudWatch 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 日志代理配置文件描述了 CloudWatch 日志代理所需的信息。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`。该文件采用 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

设置为 true（默认）时，启用 gzip http 内容编码以将压缩的负载发送到 CloudWatch 日志。这可以降低 CPU 使用率 NetworkOut、降低和减少放置延迟。要禁用此功能，请将 use_gzip_http_content_encoding = false 添加到日志代理配置文件的 [常规] 部分，然后重新启动代理。 CloudWatch

> **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 strftime() 和 strptime() 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。

file

指定要推送到 CloudWatch 日志的日志文件。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*）。只有最新的文件才会根据文件修改时间推送到 CloudWatch 日志。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。除非所有指定的行都可用，否则指纹线不会发送到 CloudWatch 日志。

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。

encoding

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。

使用带有 HTTP 代理的 CloudWatch 日志代理

您可以将 CloudWatch 日志代理与 HTTP 代理一起使用。

> **Note**
>
> awslogs-agent-setup.py 版本 1.3.8 或更高版本支持 HTTP 代理。

将 CloudWatch 日志代理与 HTTP 代理配合使用

1. 请执行以下操作之一：

   a. 要全新安装 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. 要安装现有的 Log CloudWatch s 代理，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 日志代理配置文件

如果您在 awscli-cwlogs 1.3.3 或更高版本中使用 awslogs-agent-setup .py 版本 1.3.8 或更高版本，则可以通过在 /var/awslogs/etc/config/ 目录中创建其他配置文件来为各种组件独立导入不同的流配置。当 CloudWatch Logs 代理启动时，它会在这些附加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 日志代理常见问题解答

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？

如果您使用安装脚本来安装 CloudWatch Logs 代理，则可以使用 /var/awslogs/bin/awslogs-version.sh 来检查您使用的代理版本。它会打印代理的版本及其主要依赖关系。如果你使用 yum 安装 CloudWatch 日志代理，你可以使用 “yum info awslogs” 和 “yum info aws-cli-plugin-cloudwatch-logs” 来检查 Logs 代理和插件的版本。 CloudWatch

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**
>
> 跳过数据时，Log CloudWatch s 代理会在其日志中写入警告。

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 代理可以从其停止的地方开始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 日志代理需要CreateLogGroup、CreateLogStreamDescribeLogStreams、和PutLogEvents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。

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

```
 ]
}
```

我不想让 L CloudWatch ogs 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。

使用 CloudWatch 指标进行监控

CloudWatch 日志 CloudWatch 每分钟向 Amazon 发送一次指标。

CloudWatch 记录指标

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 是一项 CloudWatch 日志服务使用量指标。有关更多信息，请参阅 CloudWatch 记录服务使用情况指标。 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，Lo CloudWatch gs 收到错误的日志事件的数量。如果目标服务返回可重试的错误，例如限制异常或可重试的服务异常（例如 HTTP 5xx），则 CloudWatch 日志会继续重试传送长达 24 小时。 CloudWatch 如果错误是不可重试的错误（例如或），则日志不会尝试重新传送。AccessDeniedException ResourceNotFoundException 有效尺寸： LogGroupName、 DestinationType、 FilterName、 PolicyLevel 有效统计数据：Sum 单位：无
DeliveryThrottling	将数据转发到订阅目标时，限制 CloudWatch 日志的日志事件数量。 如果目标服务返回可重试的错误，例如限制异常或可重试的服务异常（例如 HTTP 5xx），则 CloudWatch 日志会继续重试传送长达 24 小时。

指标	描述
	CloudWatch 如果错误是不可重试的错误（例如或），则日志不会尝试重新传送。AccessDeniedException ResourceNotFoundException 有效尺寸： LogGroupName、 DestinationType、 FilterName、 PolicyLevel 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
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。有关数据类型的信息MetricDatum ，请参阅 Amazon CloudWatch API 参考MetricDatum中的。 **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

指标	描述
ErrorCount	在您账户中执行的导致了错误的 API 操作的数量。 ErrorCount 是一项 CloudWatch 日志服务使用量指标。有关更多信息，请参阅 CloudWatch 记录服务使用情况指标。 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。 有效尺寸： LogGroupName、 DestinationType、 FilterName 有效统计数据：Sum 单位：字节
ForwardedLogEvents	转发到订阅目标的日志事件的数量。 有效尺寸： LogGroupName、 DestinationType、 FilterName、 PolicyLevel 有效统计数据：Sum 单位：无
IncomingBytes	上传到日志的日志事件量（以未压缩字节为 CloudWatch 单位）。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。 有效尺寸： LogGroupName 有效统计数据：Sum 单位：字节

指标	描述
IncomingLogEvents	上传到 CloudWatch 日志的日志事件数量。当用于 LogGroupName 维度时，这是上传到日志组的日志事件的数量。 有效尺寸： LogGroupName 有效统计数据：Sum 单位：无
LogEventsWithFindings	与您正在使用日志数据保护功能审核的数据字符串相匹配的 CloudWatch 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
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 是一项 CloudWatch 日志服务使用量指标。有关更多信息，请参阅 CloudWatch 记录服务使用情况指标。 有效维度：类、资源、服务、类型 有效统计数据：Sum 单位：无

CloudWatch 日志指标的维度

下表列出了可用于 CloudWatch 日志指标的维度。

维度	描述
LogGroupName	要显示其指标的 CloudWatch 日志日志组的名称。

维度	描述
DestinationType	CloudWatch 日志数据的订阅目标，可以是 AWS 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。订阅筛选器名称会自动转换 CloudWatch 为 ASCII，任何不支持的字符都将替换为问号 (?)。

下表列出了与账户级订阅筛选条件相关的指标的维度。

维度	描述
PolicyLevel	政策适用的级别。目前，该维度的唯一有效值是 AccountPolicy
DestinationType	CloudWatch 日志数据的订阅目标，可以是 AWS 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。订阅筛选器名称会自动转换 CloudWatch 为 ASCII，任何不支持的字符都将替换为问号 (?)。

CloudWatch 记录服务使用情况指标

CloudWatch 日志向其 CloudWatch 发送跟踪使用情况的指标 CloudWatch Logs API 操作。这些指标对应于 AWS 服务配额。跟踪这些指标可帮助您主动管理配额。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 日志服务配额的更多信息，请参阅CloudWatch 日志配额。

CloudWatch 日志每分钟都会在AWS/Usage和AWS/Logs命名空间中发布服务配额使用量指标。

下表列出了 Logs 发布的服务使用情况 CloudWatch 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。

指标

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 AWS/Logs 中。
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 AWS/Logs 中。

尺寸

维度	描述
Account metrics	使用此维度获取所有 L CloudWatch ogs API 的指标总和。 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	包含资源的 AWS 服务的名称。对于 CloudWatch 日志使用量指标，此维度的值为Logs。
Class	正在跟踪的资源类别。 CloudWatch 日志 API 使用情况指标使用此维度，其值为None。
Type	所跟踪的资源的类型。目前，当 Service 维度为 Logs 时，Type 的唯一有效值为 API。

维度	描述
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 日志配额

下表提供了 AWS 账户 CloudWatch 日志的默认服务配额，也称为限制。这些服务配额中的大多数（但不是全部）都列在 Service Quotas 控制台的 Amazon CloudWatch Logs 命名空间下。要请求提高配额，请参阅本节后面的程序。

资源	默认限额
账户级别的政策	每个账户都有一个账户级别的订阅筛选政策。 每个账户都有一个账户级别的数据保护政策。 无法更改这些限额。
异常探测器	每个账户 10 个异常检测器。无法更改此配额。
批次大小	最大批处理大小为 1,048,576 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。无法更改此配额。
数据存档	最多免费存档 5GB 的数据。无法更改此配额。
CreateLogGroup	每秒 10 笔交易（TPS/账户/区域），之后交易将被限制。您可以请求提高限额。
CreateLogStream	每秒 50 个事务（TPS/账户/区域），随后事务将被限制。您可以请求提高限额。
自定义数据标识符	每项数据保护策略最多可以包含 10 个自定义数据标识符。您可以请求提高限额。 每个定义自定义数据标识符的正则表达式最多可包含 200 个字符。无法更改此配额。
DeleteLogGroup	每秒 10 笔交易（TPS/账户/区域），之后交易将被限制。您可以请求提高限额。
DeleteLogStream	每秒 15 笔交易（TPS/账户/区域），之后交易将被限制。您可以请求提高限额。

资源	默认限额
DescribeLogGroups	每秒 10 笔交易（TPS/账户/区域）。您可以请求提高限额。
DescribeLogStreams	每秒 25 笔交易（TPS/账户/区域）。您可以请求提高限额。
发现的日志字段	CloudWatch Logs Insights 最多可以在一个日志组中发现 1000 个日志事件字段。无法更改此配额。 有关更多信息，请参阅 支持的日志和发现的字段。
从 JSON 日志中提取日志字段	CloudWatch Logs Insights 最多可以从 JSON 日志中提取 200 个日志事件字段。无法更改此配额。 有关更多信息，请参阅 支持的日志和发现的字段。
导出任务	每个账户一次一个活动（运行中或等待中）的导出任务。无法更改此配额。
FilterLogEvents	美国东部（弗吉尼亚州北部）每秒 25 个请求。 以下区域每秒 5 个请求： • 亚太地区（雅加达） • 亚太地区（大阪） • 欧洲地区（法兰克福） • 加拿大西部（卡尔加里） • 以色列（特拉维夫） 其他地区每秒 10 个请求。 无法更改此配额。

资源	默认限额
GetLogEvents	欧洲（巴黎）每秒 30 个请求。 以下区域中每秒 10 个请求： • 美国西部（俄勒冈州） • 亚太地区（雅加达） • 亚太地区（大阪） • 加拿大西部（卡尔加里） • 欧洲地区（爱尔兰） • 欧洲地区（法兰克福） • 以色列（特拉维夫） 所有其他区域每秒 25 个请求。 无法更改此配额。 如果您持续处理新的数据，建议您进行订阅。如果需要历史数据，建议将数据导出到 Amazon S3。
传入数据	最多免费传入 5GB 的数据。无法更改此配额。
Live Tail 并行会话。	15 个并行会话。您可以请求提高限额。
Live Tail：在一个会话中搜索日志组。	在一个 Live Tail 会话中最多扫描 10 个日志组。无法更改此配额。
日志事件大小	256 KB（最大）。无法更改此配额。
日志组	每个账户每个区域 1,000,000 个日志组。您可以请求提高限额。 对可属于一个日志组的日志流数没有配额。
指标筛选器	每个日志组 100 个。无法更改此配额。

资源	默认限额
嵌入式指标格式指标	每个日志事件 100 个指标，每个指标 30 个维度。有关嵌入式指标格式的更多信息，请参阅 Amazon CloudWatch 用户指南中的规范：嵌入式指标格式。
PutLogEvents	PutLogEvents 请求的最大批量大小为 1MB。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。 每个区域每个账户每秒 5000 笔交易您可以使用该服务请求提高每秒限制配额。 Service Quotas
查询执行超时值	“ CloudWatch 日志见解” 中的查询会在 60 分钟后超时。无法更改此时间限制。
查询的日志组	在单个 Logs Insights 查询中最多可以查询 50 个 CloudWatch 日志组。无法更改此配额。
查询并发性	对于标准类日志组，最多 30 个并发 Lo CloudWatch gs Insights 查询，包括已添加到仪表板的查询。 对于 Infrequent Access 类日志组，最多 5 个并发 Lo CloudWatch gs Insights 查询，包括已添加到仪表板的查询。 无法更改这些限额。
由自然语言生成的查询	多达五个并发的自然语言生成的查询请求。
查询可用性	在控制台中构建的查询可通过历史记录命令使用 30 天。此可用性期间无法更改。 使用创建的查询定义PutQueryDefinition不会过期。
查询结果的可用性	查询的结果可在 7 天内检索。无法更改此可用时间。

资源	默认限额
显示在控制台中的查询结果	控制台默认最多显示 1000 行查询结果。在查询中，您可以使用 **limit** 命令将此数量增加到最多 10000 行。有关更多信息，请参阅 CloudWatch 日志见解查询语法。
正则表达式	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。无法更改此配额。 为指标筛选条件和订阅筛选条件创建带分隔符的或 JSON 筛选条件模式，或筛选日志事件时，每个筛选条件模式最多有 2 个正则表达式。
资源策略	每个账户每个区域最多 10 个 CloudWatch 日志资源策略。无法更改此配额。
保存的查询	每个账户的每个区域最多可以保存 1000 CloudWatch 个 Logs Insights 查询。无法更改此配额。
订阅筛选器	每个日志组 2 个。无法更改此配额。

管理您的 CloudWatch 日志服务配额

CloudWatch Logs 已与 Service Quotas 集成，Service Quotas 是一项使您能够从中央位置查看和管理配额的 AWS 服务。有关更多信息，请参阅《服务限额用户指南》中的什么是服务限额？

Service Quotas 可以轻松查找 CloudWatch 日志服务配额的值。

AWS Management Console

使用控制台查看 CloudWatch 日志服务配额

1. 访问 https://console.aws.amazon.com/servicequotas/，打开服务限额控制台。
2. 在导航窗格中，选择 AWS 服务。
3. 从AWS 服务列表中，搜索并选择 Amazon CloudWatch 日志。

   在服务限额列表中，您可以查看服务限额名称、应用的值（如果该值可用）、 AWS 默认限额以及限额值是否可调整。

4. 要查看有关服务限额的其他信息（如描述），请选择限额名称。
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。

AWS CLI

要查看 CloudWatch 日志服务配额，请使用 AWS CLI

运行以下命令以查看默认 CloudWatch 日志配额。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code logs \
    --output table
```

要使用更多地使用服务配额 AWS CLI，请参阅 Ser vice Quotas AWS CLI 命令参考。要请求提高配额，请参阅 AWS CLI 命令参考中的 `request-service-quota-increase` 命令。

文档历史记录

下表描述了从 2018 年 6 月开始的《 CloudWatch 日志用户指南》每个版本中的重要更改。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 Insights 支持自然语言查询生成	CloudWatch Logs Insights 支持使用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 L CloudWatch ogs Insights 查询。	2024年6月20日
CloudWatchLogsReadOnlyAccess政策已更新	CloudWatch Logs 向添加了`cloudwatch:GenerateQuery` 权限 CloudWatchLogsReadOnlyAccess，因此拥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政策已更新	CloudWatch Logs 向添加了`cloudwatch:GenerateQuery` 权限 CloudWatchLogsFullAccess，因此拥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 日志添加了日志模式分析	CloudWatch 现在，每次您执行 Logs Insights 查询时，日志都会扫描 CloudWatch 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 日志增加了日志异常检测	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，并在日志数据中发现异常。有关更多信息，请参阅日志异常检测。	2023 年 11 月 26 日
CloudWatch 日志增加了比较功能	现在，您可以使用 CloudWatch Logs Insights 来比较日志事件在一段时间内的变化。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 日志添加了新的日志类	CloudWatch 日志支持两类日志组，因此您可以为不经常访问的日志提供经济实惠的选项，并且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。	2023 年 11 月 26 日
CloudWatch 日志见解支持自然语言查询生成	CloudWatch Logs Insights 支持使用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 L CloudWatch ogs Insights 查询。	2023 年 11 月 26 日
CloudWatch Logs 为 Live Tail 添加了正则表达式过滤模式语法支持	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 日志用户指南中的筛选模式语法。	2023 年 11 月 13 日

CloudWatch Logs 增加了对指标筛选器、订阅过滤器和筛选日志事件的正则表达式筛选模式语法支持	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 日志用户指南中的筛选模式语法。	2023 年 9 月 5 日
CloudWatch 日志见解添加了模式命令	现在，您可以在 CloudWatch Logs Insights 查询中使用模式将日志数据自动聚类为模式。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 日志用户指南中的模式。	2023 年 7 月 17 日
CloudWatch 日志见解添加了重复数据删除命令	现在，您可以在 CloudWatch Logs Insights 查询中使用重复数据删除，根据您指定的字段中的特定值删除重复的结果。有关更多信息，请参阅 Amazon L CloudWatch ogs 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 Log CloudWatch s 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 日志添加了 Live Tail 功能，因此你可以在采集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
CloudWatchLogsReadOnlyAccess政策已更新	CloudWatch 记录已添加的权限CloudWatchLogsReadOnlyAccess。添加了`logs:StartLiveTail` 和`logs:StopLiveTail` 权限，以便拥有此策略的用户可以使用控制台启动和停止 LCloudWatch ogs 实时尾部会话。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 日志见解已发布	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 日志用户指南中的使用日志见解分析 CloudWatch 日志数据	2018 年 11 月 27 日
对 Amazon VPC 终端节点的支持	现在，您可以在 VPC 和 CloudWatch 日志之间建立私有连接。有关更多信息，请参阅 Amazon CloudWatch CloudWatch 日志用户指南中的将日志与接口 VPC 终端节点配合使用。	2018 年 6 月 28 日

下表描述了《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 终端节点来防止您的 Amazon VPC 和 CloudWatch 日志之间的流量离开亚马逊网络。有关更多信息，请参阅 在接口 VPC 终端节点上使用 CloudWatch 日志。	2018 年 3 月 7 日
Route 53 DNS 查询日志	您可以使用 CloudWatch 日志来存储有关 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 日志？ 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 日志中标记 CloudWatch 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 查看发送到日志的 CloudWatch 日志数据。	2016 年 8 月 29 日
增加了对 Amazon CloudWatch 日志和新 CloudWatch 日志指标的 AWS CloudTrail 支持	增加了对 CloudWatch 日志的 AWS CloudTrail 支持。有关更多信息，请参阅 记录 CloudWatch 日志 API 和控制台操作在 AWS CloudTrail。	2016 年 3 月 10 日
增加了对将 CloudWatch 日志导出到	增加了对将 CloudWatch 日志数据导出到 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
Amazon S3 的支持		
增加了对 Amazon CloudWatch 日志中 AWS CloudTrail 记录的事件的支持	您可以在中创建警报 CloudWatch ，接收捕获的特定 API 活动的通知， CloudTrail 并使用该通知进行故障排除。	2014 年 11 月 10 日
增加了对 Amazon CloudWatch 日志的支持	您可以使用 Amazon CloudWatch Logs 来监控、存储和访问来自亚马逊弹性计算云 (Amazon EC2) 实例或其他来源的系统、应用程序和自定义日志文件。然后，您可以使用 Amazon CloudWatch 控制台、中的 CloudWatch 日志命令或日志 SDK 从 CloudWatch 日志中 AWS CLI检索关联的 CloudWatch 日志数据。有关更多信息，请参阅 什么是 Amazon CloudWatch 日志？。	2014 年 7 月 10 日

AWS 词汇表

有关最新 AWS 术语，请参阅《AWS 词汇表 参考资料》中的AWS 词汇表。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。