aws

用户指南

Amazon CloudWatch 日志



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

Amazon CloudWatch 日志: 用户指南

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

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

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 5
日志类 6
支持的功能 6
使用 Amazon L CloudWatch ogs 智能分层优化存储成本 9
CloudWatch 日志智能分层访问等级 9
构成访问权限的操作 10
启用 CloudWatch 日志智能分层 10
所需的权限 10
使用 控制台启用 11
使用启用 AWS CLI 11
查看您的当前配置 11
监控每层存储空间 12
CloudWatch 日志智能分层的定价 12
开始使用 13
先决条件 13
注册参加 AWS 账户 13
设置命令行界面 13
使用统一 CloudWatch 代理 14
使用以前的 CloudWatch 代理 14
CloudWatch 记录代理先决条件 15
快速入门：在运行的 EC2 Linux 实例上安装代理 15
快速入门：启动时在 EC2 Linux 实例上安装代理 21
快速入门：在 Windows Server 2016 实例中使用 CloudWatch 日志 25
快速入门：在 Windows Server 2012 和 Windows Server 2008 实例上使用 CloudWatch 日志 35
报告 CloudWatch Logs 代理状态 44
启动 CloudWatch 日志代理 44
停止 CloudWatch 日志代理 45
CloudWatch 日志代理参考 45

快速入门 CloudFormation 56
通过 HTTP 端点记录摄取 58
常见行为 58
不记名令牌认证 59
选项 1：使用 AWS 控制台快速入门 60
选项 2：手动设置 61
控制生成和使用 CloudWatch 日志 API 密钥的权限 64
轮换 API 密钥 66
对泄露的 API 密钥作出响应 67
API 密钥的安全最佳实践 68
使用记录 API 密钥的使用情况 CloudTrail 69
OTLP 端点 70
请求格式 70
示例请求 71
响应 72
OTLP 特有的行为 72
HLC 终端节点 72
输入模式 72
事件字段（必填） 73
时间字段（可选） 73
Content-Type 74
接受的 JSON 值类型 74
终端节点格式 74
请求格式 75
示例请求 76
最佳实践 76
限制 77
ndjson 端点 77
请求格式 77
接受的 JSON 值类型 78
时间戳字段 79
无效的行 79
示例请求 80
响应 80
最佳实践 81
限制 81

结构 JSON 端点 81
请求格式 82
接受的 JSON 值类型 82
时间戳字段 83
示例请求 83
响应 83
最佳实践 84
限制 85
HTTP 摄取端点的比较 85
选择终端节点 86
系统日志摄取 87
支持的协议和端口 87
支持的系统日志格式 87
RFC 5424 提取的字段 88
RFC 3164 提取的字段 89
思科 FTD/ASA 提取的字段 89
消息传输 90
限额和限制 90
设置 91
先决条件 91
步骤 1：创建或标识 VPC 92
步骤 2：创建安全组 94
步骤 3：创建 VPC 端点 95
步骤 4：创建日志组 96
步骤 5：添加资源策略 97
步骤 6：创建系统日志配置 98
第 7 步：验证交付和字段提取 99
设备配置 100
rsyslog 101
syslog-ng 102
快速验证 104
网络设备 104
TLS 证书信任 105
监控 105
系统日志指标 105
掉线原因 106

连接被拒绝的原因 106
建议的警报 107
在自动仪表板上查看 syslog 指标 107
示例查询 108
安全和访问监控 108
运营可见性 109
防火墙和网络设备查询 110
VPC 端点策略 110
支持的条件键 111
限制到特定的日志组 112
拒绝所有流量 112
重置为默认策略 113
策略传播 113
问题排查 113
消息未出现在日志组中 113
TLS 握手失败 114
UDP 消息丢失 115
连接被拒绝 115
由于速率限制而丢弃了邮件 115
使用 AWS 软件开发工具包 117
日志管理 119
数据源发现和管理 119
什么是 CloudWatch 日志数据源？ 119
如何开始 120
系统字段 121
访问数据源 122
与日志组的关系 122
数据源启用的功能 122
支持的数据来源 123
支持 AWS 服务 用于数据源 123
数据源支持的第三方来源 128
自定义来源 145
使用日志分析分析 CloudWatch 日志数据 146
你可以用日志分析做什么 146
开始使用日志分析 146
使用 “日志见解” 分析 CloudWatch 日志数据 147

支持的查询语言 149
CloudWatch 日志见解查询语言（Logs Insights QL） 150
OpenSearch 管道处理语言 (PPL) 231
OpenSearch 结构化查询语言 (SQL) 240
使用自然语言生成和更新 L CloudWatch ogs Insights 查询 249
示例查询 250
选择不使用您的数据来改进服务 252
支持的日志和发现的字段 253
JSON 日志中的字段 255
创建字段索引以提高查询性能并减少扫描量 257
自动索引字段 260
字段索引语法和配额 261
创建账户级字段索引策略 266
创建日志组级字段索引策略 267
创建查询时的日志组选择选项 267
删除字段索引策略的影响 268
使用分面对日志进行分组和浏览 268
运行基于分面的查询 269
保存基于分面的查询 270
创建账户级分面 270
使用 API 进行分面管理 270
在“日志见解”中查看周围的 CloudWatch 日志 271
周围日志的工作原理 271
查看周围的日志 271
在周围的日志中搜索 272
模式分析 273
开始使用模式分析 274
有关 pattern 命令的详细信息 276
保存并重新运行查询 276
使用带参数的已保存查询 282
将查询添加到控制面板或导出查询结果 284
查看正在运行的查询或查询历史记录 285
使用加密查询结果 AWS Key Management Service 285
限制 286
步骤 1：创建一个 AWS KMS key 286
步骤 2：设置 KMS 密钥的权限 287

步骤 3：将 KMS 密钥与查询结果关联 288
步骤 4：取消密钥与账户中查询结果的关联 288
根据 Lo CloudWatch gs Insights 查询结果生成自然语言摘要 289
工作原理 289
区域可用性和数据处理 289
开始使用 290
Permissions 290
数据隐私 290
使用计划查询自动进行日志分析 291
了解计划查询的概念 291
IAM 角色分离 291
Cross-region 以及跨账户的使用 294
调度表达式和时区处理 295
选择查询语言 295
目的地选择和用例 296
查询结果的格式和结构 297
了解已扫描的记录、匹配的记录和结果数 299
时间表表达式参考 301
最佳实践 305
计划查询入门 306
创建计划查询 306
查看和管理预定查询 310
查看预定查询执行历史记录 311
更新预定查询 313
为计划查询配置 S3 目标 315
将结果传送到同一账户中的 Amazon S3 存储桶 315
将结果传送到另一个账户中的 Amazon S3 存储桶 316
由客户管理对结果进行加密 AWS KMS 键 317
为计划查询配置查找表目的地 319
对计划查询进行故障排除 320
查询执行因权限错误而失败 320
查询超时 321
目标处理失败 321
无效的查询错误 321
查询并发错误 322
日志异常检测 323

异常和模式的严重性和优先级 324
异常可见性时间 324
抑制异常 324
常见问题 324
在“ CloudWatch 日志见解”中使用异常检测 325
对日志组启用异常检测 326
查看已发现的异常 327
在日志异常检测器上创建警报 329
日志异常检测器发布的指标 331
使用对异常检测器及其结果进行加密 AWS KMS 331
限制 331
使用 L CloudWatch ogs Live Tai 336
使用 Live Tail 开始直播会话 AWS CLI 336
print-only 337
interactive 337
在控制台中启动 Live Tail 会话 339
Cross-account 跨区域日志集中化 342
数据集中化概念 342
设置日志集中化 343
先决条件 343
自定义目标日志组名称 346
创建集中化规则 347
修改集中化规则 350
查看集中化规则 350
删除集中化规则 350
监控和排查集中化规则 351
集中化规则运行状况 351
使用监控集中化 API 调用 AWS CloudTrail 352
监控建议 352
使用日志组和日志流 355
创建日志组 355
将日志发送到日志组 355
查看日志数据 356
使用筛选条件模式搜索日志数据 356
使用控制台搜索日志条目 357
使用搜索日志条目 AWS CLI 357

从指标定向至日志 358
问题排查 358
更改日志数据保留期 359
保护日志组不被删除 360
保护日志组不被删除 360
启用删除保护 360
标记日志组 361
有关标签的基本知识 361
使用标签跟踪成本 362
标签限制 362
使用标记日志组 AWS CLI 363
使用日志 API 标记 CloudWatch 日志组 363
使用加密日志数据 AWS KMS 363
限制 364
步骤 1：创建 AWS KMS 密钥 286
步骤 2：设置 KMS 密钥的权限 365
步骤 3：为调用方的 IAM 委托人设置权限 368
步骤 4：将 KMS 密钥与日志组关联 335
步骤 5：取消密钥与日志组的关联 335
KMS 密钥和加密上下文 371
通过屏蔽帮助保护敏感的日志数据 374
了解数据保护策略 377
创建或使用数据保护策略所需的 IAM 权限 379
创建账户范围的数据保护策略 384
为单个日志组创建数据保护策略 387
查看未屏蔽的数据 390
审计结果报告 390
您可以保护的数据类型 392
在摄取期间转换日志 431
创建和管理日志转换器 432
创建账户级转换器策略 433
编辑或删除账户级转换器策略 434
从头开始创建对 log-group-level数转换器 435
通过复制现有的 log-group-level变压器来创建变压器 436
编辑变 log-group-level压器 436
删除变 log-group-level压器 437

可配置的解析器类型处理器 438
parseJSON 438
grok 439
csv 469
parseKeyValue 471
用于 AWS 出售日志的内置处理器 473
parseWAF 473
parsePostgres 476
parseCloudfront 477
parseRoute53 478
parseVPC 479
parseToOCSF 480
字符串变异处理器 482
lowerCaseString 482
upperCaseString 483
splitString 484
substituteString 486
trimString 489
JSON 变异处理器 490
addKeys 491
deleteKeys 492
moveKeys 493
renameKeys 495
copyValue 496
listToMap 498
数据类型转换器处理器 503
typeConverter 503
datetimeConverter 504
转换指标和错误 506
使用 S3 表格集成访问日志 507
了解 S3 表集成 507
核心组件 507
数据流向 S3 表 507
何时使用 S3 表格集成 508
先决条件 508
IAM 权限 508

KMS 密钥策略（适用于加密数据） 510
开始使用 511
将亚马逊 Amazon S3 表格与 AWS 分析服务-使用 Amazon S3 控制台 (Link) 512
配置 Lake Formation 权限 513
Connect 使用分析工具 513
指标筛选条件 515
概念 516
指标筛选条件的筛选条件模式语法 517
为指标筛选条件配置指标值 518
使用日志事件中的指标发布维度 519
使用录入事件中的值来增加指标的值 522
创建指标筛选条件 523
创建日志组的指标筛选条件 523
示例：对日志事件进行计数 524
示例：对字词的出现次数进行计数 525
示例：对 HTTP 404 代码进行计数 527
示例：对 HTTP 4xx 代码进行计数 529
示例：从 Apache 日志中提取字段并分配维度 531
列出指标筛选条件 532
删除指标筛选条件 533
订阅筛选条件 535
概念 536
日志组级别订阅筛选条件 538
示例 1：使用 Amazon Kinesis 数据流的订阅筛选条件 538
示例 2：订阅筛选条件为 AWS Lambda 544
示例 3：Amazon Data Firehose 订阅筛选条件 547
示例 4：亚马逊 OpenSearch 服务的订阅筛选条件 554
Account-level 订阅过滤器 555
示例 1：使用 Amazon Kinesis 数据流的订阅筛选条件 556
示例 2：订阅筛选条件为 AWS Lambda 562
示例 3：Amazon Data Firehose 订阅筛选条件 566
Cross-account 跨区域订阅 573
Cross-account 使用 Amazon Kinesis 数据流进行跨区域日志数据共享 574
Cross-account 使用 Firehose 进行跨区域日志数据共享 592
Cross-account 使用 Amazon Kinesis 数据流进行跨区域账户级订阅 605
Cross-account 使用 Firehose 进行跨区域账户级别订阅 622

混淆代理问题防范 633
防止日志递归 634
筛选条件模式语法 636
支持的正则表达式 636
使用正则表达式匹配字词 639
匹配非结构化日志事件中的字词 639
匹配 JSON 日志事件中的字词 643
匹配以空格分隔的日志事件中的字词 651
启用从的日志记录 AWS 务 655
支持的日志目标和权限 658
需要额外权限 [V1] 的日志记录 672
发送到日志的 CloudWatch 日志 673
发送到 Amazon S3 的日志 676
日志发送至 Firehose 680
需要额外权限 [V2] 的日志记录 682
日志传送设置示例 683
发送到日志的 CloudWatch 日志 686
发送到 Amazon S3 的日志 690
日志发送至 Firehose 696
追踪已发送至 X-Ray 699
Service-specific 权限 702
Console-specific 权限 703
Cross-account 配送示例 704
创建传输源 705
配置传输到 Amazon S3 存储桶 705
配置向 Firehose 流的传输 709
Cross-service 混乱的副手预防 711
使用遥测启用规则自动启用日志 712
规则评估层次结构 712
创建启用规则 712
启用来自第三方来源的日志记录 713
直接第三方集成 713
通过 S AWS ecurity Hub CSPM 提供的其他来源 713
将日志数据导出到 Amazon S3 719
概念 720
使用控制台将日志数据导出到 Amazon S3 721

Same-account 导出（控制台） 721
Cross-account 导出（控制台） 727
使用将日志数据导出到 Amazon S3 AWS CLI 736
Same-account 导出 (CLI) 737
Cross-account 导出 (CLI) 743
描述导出任务 (CLI) 752
取消导出任务 (CLI) 754
将数据流式传输到 OpenSearch 服务 755
先决条件 755
为日志组订阅 OpenSearch 服务 755
代码示例 758
基本功能 759
你好 CloudWatch 日志 760
操作 761
场景 835
配置容器服务连接 836
创建您的第一个无服务器函数 852
学习 syslog 提取基础知识 864
运行大型查询 875
使用计划的事件调用 Lambda 函数 913
使用加密查找表 AWS KMS 916
CloudWatch 日志如何 AWS KMS 用于查找表 916
所需的权限 917
步骤 1：创建 AWS KMS 密钥 917
步骤 2：设置 KMS 密钥的权限 918
步骤 3：将 KMS 密钥与查找表关联 919
注意事项 920
安全性 921
数据保护 921
静态加密 922
传输中加密 922
Identity and access management 923
身份验证 923
访问控制 923
有关管理访问的概述 924
使用基于身份的策略（IAM 策略） 928

合规性验证 953
恢复能力 953
基础结构安全性 953
接口 VPC 端点 954
可用性 954
为 CloudWatch 日志创建 VPC 终端节点 954
测试您的 VPC 和 CloudWatch 日志之间的连接 954
控制对您的 CloudWatch 日志 VPC 终端节点的访问权限 955
对 VPC 上下文键的支持 956
使用记录 API 和控制台操作 AWS CloudTrail 957
查询生成信息 CloudTrail 961
了解日志文件条目 962
使用 CloudWatch 指标监控使用情况 964
CloudWatch 记录指标 964
CloudWatch 日志指标的维度 968
日志转换器指标和维度 969
集中化指标和维度 970
CloudWatch 记录服务使用情况指标 973
服务配额 975
管理您的 CloudWatch 日志服务配额 985
文档历史记录 987
AWS 词汇表 999
..... m

什么是 Amazon CloudWatch 日志？

您可以使用 Amazon Lo CloudWatch gs 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。
- 创建字段索引以提高查询效率：您可以为日志事件中的字段创建字段索引。然后，当您在 L CloudWatch ogs Insights 查询中使用字段索引时，该查询会尝试跳过处理已知不包含索引字段的日志事件。此查询可减少查询的扫描量，从而更快地返回结果。要开始使用，请参阅创建字段索引以提高查询性能并减少扫描量。
- 使用 Live Tail 进行检测和调试 — 您可以使用 Live Tail 通过查看提取的新日志事件的流式列表来快速排查事件。您可以近乎实时地查看、筛选和突出显示提取的日志，帮助您快速检测并解决问题。您可以根据指定的术语筛选日志，也可以突出显示包含指定术语的日志，以帮助快速找到所需内容。有关更多信息，请参阅 使用 L CloudWatch ogs Live Tai。

- 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。
- 删除保护-一种防止意外删除日志组及其日志流的保护措施。在日志组上启用删除保护后，删除保护会阻止所有删除操作，直到它被明确禁用。默认情况下，删除保护未启用。此可选功能有助于保护关键的操作和合规性数据免遭意外删除，例如包含审计数据的日志组以及用于故障排除和分析的生产应用程序日志。
- 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。
- 跨账户和区域集中日志 — 您可以使用 CloudWatch 日志集中来定义跨账户和跨区域集中化规则，将跨多账户和区域环境摄取的日志数据复制到中心区域和账户。您可以配置中央账户内的备份区域以提高恢复能力，为中央账户中新创建的日志组定义加密设置，并针对来自特定源账户和区域的日志事件创建指标和订阅筛选条件，从而增强筛选能力。

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

日志事件

日志事件是对受监视的应用程序或资源记录的一些活动的记录。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条件从已接收的事件中提取指标观察数据并将它们转换为 CloudWatch 指标中的数据点。指标筛选条件将分配给日志组，分配给日志组的所有筛选条件都将应用于其日志流。

保留期设置

保留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。

删除保护

删除保护是一种保护措施，可防止意外删除日志组及其日志流。在日志组上启用删除保护后，删除保护会阻止所有删除操作，直到它被明确禁用。默认情况下，删除保护未启用。此可选功能有助于保护关键的操作和合规性数据免遭意外删除，例如包含审计数据的日志组以及用于故障排除和分析的生产应用程序日志。

Amazon CloudWatch Logs 账单和费用

有关如何分析 CloudWatch 日志的成本和使用量的详细信息 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。

有关定价的更多信息，请参阅 Amazon CloudWatch 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）。

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**
>
> 对于费用，标准和不频繁访问日志类仅在提取成本方面有所不同。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**
>
> 在创建日志组后，无法更改其日志类。

支持的功能

下表列出了每个日志类的功能。

功能	标准	不频繁访问	
完全托管日志提取和存储	是 ✓	是 ✓	
跨账户功能	是 ✓	是的 ✓	
使用加密 AWS KMS	是的 ✓	是的 ✓	

功能	标准	不频繁访问	
CloudWatch 日志见解查询命令	是的 ✓	是 ✓（大多数命令 – 请参阅 日志类中支持的 Logs Insights QL 命令。）	
CloudWatch 日志见解已发现字段	是的 ✓	是的 ✓	
Facets	是的 ✓	否	
使用 CloudWatch 管道转换日志	是的 ✓	是的 ✓	
导出到亚马逊 S3	是的 ✓	是的 ✓	
S3 表格集成	是的 ✓	是的 ✓	
预定查询	是的 ✓	是的 ✓	
使用 OpenSearch PPL 或 OpenSearch SQL 在 Logs Insight CloudWatch s 中进行查询；	是的 ✓	是的 ✓	
自然语言查询助手	是 ✓	否	
CloudWatch 日志异常检测	是的 ✓	否	
Live Tail	是 ✓	否	
字段索引	是的 ✓	否	
与之前的时间范围比较	是 ✓	否	
订阅筛选条件	是 ✓	否	

功能	标准	不频繁访问
GetLogEvents和 FilterLogEventsAPI 操作	是的 ✓	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。
指标筛选条件	是 ✓	否
Container Insights 日志提取	是 ✓	否
Lambda Insights 日志提取	是 ✓	否
通过屏蔽保护敏感数据	是的 ✓	是的 ✓
嵌入式指标格式	是 ✓	否

> **Note**
>
> 除了这两个日志类之外，还有一个`Delivery`日志类。仅将`Delivery`日志类用于传送要存储在 Amazon S3 或中的 AWS Lambda 日志 Amazon Data Firehose。Delivery 类的日志组中的 CloudWatch Logs 日志事件会保留两天。此保留期是固定的，不能更改。此日志类不提供丰富的 CloudWatch Logs 功能，例如 CloudWatch Logs Insights 查询。

使用 Amazon L CloudWatch ogs 智能分层优化存储成本

Amazon L CloudWatch ogs 智能分层根据访问模式自动将您的日志数据分为三个访问层级：标准、不频繁访问和存档即时访问。借助智能分层，您可以以较低成本的层级保留大容量、长期保留的日志，而无需任何运营开销。您可以将它们原生保存在日志中，无论您的数据位于哪个层，都可享受相同的查询体验，而不必筛选 CloudWatch 日志或将其卸载到单独的存储解决方案。

CloudWatch 日志监控访问模式，并自动将 30 天未访问的数据重新分类为不频繁访问层，将 90 天内未访问的数据重新分类为存档即时访问层。当您访问较旧的数据时， CloudWatch Logs 会自动将其升级回标准级别，为期 30 天。通过将所有日志整合到 CloudWatch 日志中，您可以在一个工具中获得全面的可见性，从而消除管理多个存储解决方案的运营开销，并通过在一个地方查询、分析和警报所有日志来缩短平均解决时间 (MTTR)。

要开始优化存储成本，请使用 CloudWatch 控制台或通过调`PutStorageTierPolicy`用在区域内的账户级别启用智能分层。启用智能分层后，它将应用于该区域内您账户中的所有日志组。

Amazon L CloudWatch ogs 智能分层服务适用于除中东（巴林）和中东（阿联酋）地区之外的所有 AWS 商业区域。

CloudWatch 日志智能分层访问等级

无论您的日志数据位于哪个访问层，您的查询体验都将保持不变。

标准（默认）

这是默认套餐。所有新摄取的日志数据都从标准层开始。在主动访问日志数据时（自上次访问起 0 到 30 天），日志数据仍处于标准层。标准层为查询、导出、实时跟踪和日志检索提供低延迟和高吞吐量性能。

不频繁访问

如果连续 30 天未访问日志数据，则会移至不经常访问级别。该层存储 31 到 90 天前最后一次访问的数据。不频繁访问层提供与标准层相同的查询延迟和吞吐量，但存储成本较低。

存档即时访问

如果连续 90 天未访问日志数据，则会将其移至存档即时访问层。该层包含上次访问时间超过 90 天的数据。存档即时访问层以最低的存储成本提供与其他层相同的毫秒查询延迟。

构成访问权限的操作

以下访问模式会自动将日志数据从不频繁访问层或存档即时访问层提升回标准层。当这些操作中的任何一个读取较低成本级别的日志事件时，这些日志事件将移回标准级别，为期 30 天。每次后续访问时，计时器都会重置。

> 用于日志集中化的层级分类
>
> 通过跨账户跨区域日志集中化访问目标账户中的复制日志只会提升目标账户中的日志数据。

- 带查询或警报配置的出口 -使用`StartQuery`或`CreateScheduledQuery`通过 Logs Insights 查询 CloudWatch 日志数据或在日志上配置警报
- 筛选和检索日志事件 -使用`FilterLogEvents`或读`GetLogEvents`取日志事件
- 导出 — 使用以下命令将日志数据导出到 Amazon S3 `CreateExportTask`

启用 CloudWatch 日志智能分层

所需的权限

要管理账户的存储层，您需要以下 IAM 权限：

- `logs:PutStorageTierPolicy`— 需要为您的账户设置或更改存储层级。
- `logs:GetStorageTierPolicy`— 检索当前配置的存储层是必需的。还需要从控制台启用智能分层。

拥有`CloudWatchLogsFullAccess`托管策略的用户已经拥有这两个权限。拥有`CloudWatchLogsReadOnlyAccess`托管策略的用户拥有`logs:GetStorageTierPolicy`权限。

以下示例显示了授予这两种权限的策略声明：

```
{
    "Effect": "Allow",
    "Action": [
        "logs:PutStorageTierPolicy",
        "logs:GetStorageTierPolicy"
    ],
```

```
    "Resource": "*"
}
```

使用 控制台启用

要使用 CloudWatch 控制台启用智能分层，请执行以下操作：

1. 在处打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择设置。
3. 在“设置”页面上，选择“日志”选项卡。
4. 在智能分层部分中，选择启用。

使用启用 AWS CLI

要启用智能分层，请使用以下`put-storage-tier-policy`命令：

```
aws logs put-storage-tier-policy --storage-tier INTELLIGENT_TIERING
```

该命令返回已设置的存储层及其上次更新的时间：

```
{
    "storageTier": "INTELLIGENT_TIERING",
    "lastUpdatedTime": 1751328000000
}
```

要禁用智能分层，请将存储层重新设置为：`STANDARD`

```
aws logs put-storage-tier-policy --storage-tier STANDARD
```

当您将存储层重新设置为时`STANDARD`，您账户中的所有日志组都将按标准存储费率计费。

查看您的当前配置

要查看您账户的当前存储等级，请使用以下`get-storage-tier-policy`命令：

```
aws logs get-storage-tier-policy
```

该命令返回当前存储层及其上次更新的时间：

```
{
    "storageTier": "INTELLIGENT_TIERING",
    "lastUpdatedTime": 1751328000000
}
```

监控每层存储空间

启用智能分层后，L CloudWatch ogs 会发布一个销售指标，该指标显示每个日志组在每个层中存储的数据量：

- 命名空间：AWS/Logs
- 指标名称：StoredBytes
- 尺寸：LogGroupName，StorageType（值：Standard，Infrequent_Access，Archival_Instant_

您可以使用此指标来创建 CloudWatch 仪表板，显示每层存储明细、设置层大小警报或跟踪一段时间内的成本优化。

> **Note**
>
> 该指标每天发布一次，可能无法实时反映等级变化。

CloudWatch 日志智能分层的定价

使用智能分层，您可以根据数据所在的层级支付不同的存储费率。启用智能分层不收取额外费用，层级过渡不收取任何费用。您只需为每个层级消耗的存储空间付费。

有关定价的更多信息，请参阅亚马逊 CloudWatch 定价。

要查看每层存储成本，请使用 Athena 并按TimedStorage-ByteHrs（标准）、（不频繁访问）和TimedStorage-IA-ByteHrsTimedStorage-AIA-ByteHrs（存档即时访问）使用类型进行筛选。有关账单中 CloudWatch 日志使用类型的更多信息，请参阅分析、优化和降低 CloudWatch 成本。

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： CloudFormation 用于开始使用 CloudWatch 日志

先决条件

要使用亚马逊 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，您可以安装和配置 AWS Command Line Interface (AWS CLI)。

注册参加 AWS 账户

要开始使用 AWS，你需要一个 AWS 账户. 有关创建的信息 AWS 账户，请参阅《AWS 账户管理 参考指南》 AWS 账户中的 “入门” 。

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 L CloudWatch ogs 代理配置文件并设置 CloudWatch代理以收集相同的日志。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

- 报告 CloudWatch Logs 代理状态
- 启动 CloudWatch 日志代理
- 停止 CloudWatch 日志代理
- CloudWatch 日志代理参考

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，而不是较旧的 CloudWatch 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，较旧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，而不是较旧的 CloudWatch 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. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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。有关详细信息，请参阅《Amazon EC2 用户指南》中的连接到您的实例。

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

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

```
sudo yum update -y
```

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

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

4. 编辑 /etc/awslogs/awslogs.conf 文件以配置要跟踪的日志。有关编辑此文件的更多信息，请参阅 CloudWatch 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. 在代理运行一段时间后，可以在 CloudWatch 控制台看到新创建的日志组和日志流。

有关更多信息，请参阅 查看发送到日志的 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。有关详细信息，请参阅《Amazon EC2 用户指南》中的连接到您的实例。

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

2. 使用两个选项之一运行 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	说明
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。

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

   有关更多信息，请参阅 查看发送到日志的 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. 使用 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 策略。
4. 在 Review Policy（查看策略）页面上，为 Policy Name（策略名称）键入策略的名称。
5. 对于 Policy Document（策略文档），粘贴以下策略：

   JSON

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

6. 选择 Create Policy（创建策略）。
7. 在导航窗格中，选择 Roles（角色）和 Create New Role（创建新角色）。
8. 在 Set Role Name（设置角色名称）页面上，请键入角色名称，然后选择 Next Step（下一步）。
9. 在 Select Role Type（选择角色类型）部分，选择 Amazon EC2 旁的 Select（选择）。

10. 在 Attach Policy（附加策略）页面的表标题中，依次选择 Policy Type（策略类型）和 Customer Managed（客户托管）。
11. 选择您创建的 IAM policy，然后选择 Next Step（下一步）。
12. 选择创建角色。

    有关用户和策略的更多信息，请参阅 IAM 用户指南中的 IAM 用户和组以及管理 IAM policy。

启动新实例并启用 CloudWatch 日志

1. 打开位于 https://console.aws.amazon.com/ec2/ 的 Amazon EC2 控制台。
2. 选择启动实例。

   有关更多信息，请参阅《Amazon EC2 用户指南》中的启动实例。
3. 在步骤 1：选择亚马逊机器映像（AMI）页面上，选择要启动的 Linux 实例类型，然后在步骤 2：选择实例类型页面上，选择下一步：配置实例详细信息。

   确保亚马逊机器映像（AMI）中包含了 cloud-init。亚马逊 Linux AMIs 以及 AMIs 适用于 Ubuntu 和 RHEL 的 cloud-init 已经包含了 cloud-init，但是 CentOS 和其他可能不是。 AMIs 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://amzn-s3-demo-bucket/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。我们建议您使用较新的统一 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。

Steps

- 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。

Steps

- 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 添加具有唯一性 IDs（例如“CloudWatchLogs2” 和 CloudWatchLogs “3”）和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 控制台的日志组 CloudWatch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《Amazon EC2 用户指南》中的将 IAM 角色附加到实例。

验证 Systems Manager 的先决条件

在使用 Systems Manager 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. 打开位于 https://console.aws.amazon.com/ec2/ 的 Amazon EC2 控制台。
2. 打开 SSM 控制台，网址为。https://console.aws.amazon.com/systems-manager/
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 Systems Manager 先决条件。
7. 对于 Status（状态），选择 Enabled（已启用）。
8. 对于 Properties（属性），请复制并粘贴您在之前任务中创建的 JSON 内容。
9. 填写剩余选填字段并选择 Run（运行）。

执行以下程序查看 Amazon EC2 控制台中的命令执行结果。

在控制台中查看命令输出

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

快速入门：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。我们建议您使用较新的统一 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。

Steps

- 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 添加具有唯一性 IDs（例如“CloudWatchLogs2” 和 CloudWatchLogs “3”）和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 控制台的日志组 CloudWatch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 EC2 Config 服务 (EC2Config.exe)。您的实例应该有 EC2 Config 4.0 或更高版本，您可以使用此过程。

CloudWatch 使用 EC2 Config 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。

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。有关详细信息，请参阅《Amazon EC2 用户指南》中的连接到您的实例。

   有关连接问题的更多信息，请参阅《Amazon EC2 用户指南》中的排查实例的连接问题
2. 在命令提示窗口中，键入以下命令：

   ```
   sudo service awslogs status
   ```

   如果您运行的是 Amazon Linux 2，请键入以下命令：

   ```
   sudo service awslogsd status
   ```
3. 检查/var/log/awslogs.log 文件中是否存在与 CloudWatch 日志代理有关的所有错误、警告或问题。

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。有关详细信息，请参阅《Amazon EC2 用户指南》中的连接到您的实例。

   有关连接问题的更多信息，请参阅《Amazon EC2 用户指南》中的排查实例的连接问题。
2. 在命令提示窗口中，键入以下命令：

   ```
   sudo service awslogs start
   ```

   如果您运行的是 Amazon Linux 2，请键入以下命令：

   ```
   sudo service awslogsd start
   ```

停止 CloudWatch 日志代理

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。有关详细信息，请参阅《Amazon EC2 用户指南》中的连接到您的实例。

   有关连接问题的更多信息，请参阅《Amazon EC2 用户指南》中的排查实例的连接问题。
2. 在命令提示窗口中，键入以下命令：

   ```
   sudo service awslogs stop
   ```

   如果您运行的是 Amazon Linux 2，请键入以下命令：

   ```
   sudo service awslogsd stop
   ```

CloudWatch 日志代理参考

> **Important**
>
> 本节仅供那些使用已弃用的 Log CloudWatch s 代理的用户参考。如果您使用的是实例元数据服务版本 2 (imdsv2)，则必须使用新的统一代理。 CloudWatch 但是，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，而不是已弃用 CloudWatch 的

Logs 代理。有关更新的统一 CloudWatch 代理的信息，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 有关从已弃用的 CloudWatch Logs 代理迁移到统一代理的信息，请使用向导创建 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 '.'（时期）。

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。

文件

指定要推送到 CloudWatch 日志的日志文件。文件可以指向一个或多个文件（使用通配符，例如/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。

编码

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

> **Warning**
>
> 指定错误的编码可能导致数据丢失，因为无法解码的字符将被其他字符替代。

以下是一些常见编码：

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

buffer_duration

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

batch_count

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

batch_size

指定批处理中的日志事件的最大大小（以字节为单位，最大为 1048576 字节）。默认值为 1048576 字节。此大小是按中 UTF-8所有事件消息的总和加上每个日志事件的 26 字节计算得出的。

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**
>
> 在 awslogs-agent-setup.py 版本 1.3.8 或更高版本中支持 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 CloudWatch Logs 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//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。var/awslogs etc/config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。[常规] 部分中的配置属性必须在主配置文件 (//etc/awslogs.conf) 中定义，并且在 var/awslogs ///中的任何其他配置文件中都将被忽略。var/awslogs etc/config

如果您因为使用 rpm 安装代理而没有 etc/config//var/awslogs/目录，则可以改用/etc/awslogs/config/ 目录。

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 被重命名为 //syslog.log. var/log 1。如果 var/log //syslog.log.1 在之前的轮换中已经存在，则将其重命名为 //syslog.log.2。var/log
- 在创建副本后截断原始日志文件。例如，/var/log/syslog.log 被复制到/var/log/syslog.log.1，// syslog.log 被截断。var/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 仍然存在，// syslog.log.2014-01-02 已创建。var/log

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，如果每条日志消息的第一行都包含像 '2014-01-02 'T13:13:01Z这样的时间戳，则可以将 multi_line_start_patter n 设置为'\ d {4}-\ d {2}-\ d {2} T\ d {2}:\ 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 loss/duplicates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 日志代理需要权限才能执行CreateLogGroup、CreateLogStream、DescribeLogStreamsDescribeLogGroups、PutLogE作。如果您使用最新的代理，则不需要 DescribeLogStreams。请参阅以下 IAM 策略示例。

JSON

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

我不想让 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。

快速入门： 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 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 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。

通过 HTTP 端点记录摄取

Amazon CloudWatch Logs 提供了 HTTP 终端节点，允许您使用简单的 HTTP POST 请求将 CloudWatch 日志直接发送到日志。这些端点支持 Sigv4 和持有者令牌身份验证。

> **Important**
>
> 我们建议对所有可以集成 S AWS DK 的生产工作负载使用 Sigv4 身份验证。Sigv4 使用短期凭证并提供最强的安全状态。不记名令牌（API 密钥）身份验证适用于 Sigv4 不可行的场景，例如不支持 SDK 集成的第三方日志转发器。 AWS 有关更多信息，请参阅 IAM 用户指南中的长期访问密钥替代方案。

CloudWatch 日志支持以下 HTTP 摄取终端节点：

端点	路径	Content-Type	Format
OpenTelemetry Logs	`/v1/logs`	`application/json` 或 `application/x-protobuf`	OTLP JSON 或 Protobuf
HLC Logs	`/services/collector/event`	`application/json`	HLC 格式
ND-JSON Logs	`/ingest/bulk`	`application/json` 或 `application/x-ndjson`	换行符分隔的 JSON
Structured JSON Logs	`/ingest/json`	`application/json`	JSON 对象或数组

常见行为

所有 HTTP 摄取端点都具有以下行为：

身份验证

所有端点都支持 Sigv4 和持有者令牌身份验证：

- Sigv4（推荐）— 标准 AWS 签名版本 4 签名。只要您的应用程序或基础设施支持 S AWS DK 或可以签署请求，就使用 Sigv4。Sigv4 使用短期凭证，是最安全的身份验证方法。
- 不记名令牌-使用`Authorization: Bearer <ACWL token>`标题。
  - 令牌必须是有效的 ACWL 持有者令牌。有关设置说明，请参阅the section called “不记名令牌认证”。
  - 需要`logs:PutLogEvents`和 `logs:CallWithBearerToken` IAM 权限。

日志组和日志流

- 通过标题提供：`x-aws-log-group`和 `x-aws-log-stream`
- 除了 OTLP 之外，所有端点也`?logGroup=<name>&logStream=<name>`都支持查询参数。
- 不能对同一个参数同时使用查询参数和标头。
- 日志组和日志流都是必需的。

响应

- 成功：`HTTP 200`用身体 `{}`
- 验证错误：`HTTP 400`
- 身份验证失败：`HTTP 401`

设置不记名令牌身份验证

在使用持有者令牌身份验证向任何 HTTP 摄取端点发送日志之前，您需要：

- 创建具有 CloudWatch 日志权限的 IAM 用户
- 生成特定于服务的凭证（持有者令牌）
- 创建日志组和日志流
- 在日志组上启用不记名令牌身份验证

> **Important**
>
> 我们建议在可能的情况下对所有工作负载使用带有短期凭证的 Sigv4 身份验证。SigV4 提供了最强的安全态势。将 API 密钥（不记名令牌）的使用限制在基于凭证的短期身份验证不可行的

场景中。当您准备好将 CloudWatch 日志整合到具有更高安全要求的应用程序中时，应切换到短期凭证。有关更多信息，请参阅 IAM 用户指南中的长期访问密钥替代方案。

选项 1：使用 AWS 控制台快速入门

AWS 管理控制台简化了生成用于 HTTP 端点访问的 API 密钥的工作流程。

使用控制台设置 HTTP 端点访问权限

1. 登录到 AWS 管理控制台。
2. 导航至 CloudWatch> 设置 > 日志。
3. 在 API 密钥部分中，选择生成 API 密钥。
4. 对于 API 密钥过期时间，请执行以下操作之一：
   - 将 API 密钥的过期时间选择为 1、5、30、90 或 365 天。
   - 选择自定义持续时间以指定自定义 API 密钥到期日期。
   - 选择永不过期（不推荐）。
5. 选择生成 API 密钥。

   控制台自动：
   - 创建具有适当权限的新 IAM 用户
   - 附加CloudWatchLogsAPIKey访问管理策略（包括`logs:PutLogEvents`和`logs:CallWithBearerToken`权限）
   - 生成特定于服务的凭证（API 密钥）
6. 复制并安全保存显示的凭据：
   - API 密钥 ID（特定于服务的凭证 ID）
   - API 密钥密钥（持有者令牌）

> **Important**
>
> 立即保存 API 密钥密钥。之后无法检索。如果您丢失了密钥，则需要生成一个新的 API 密钥。

7. 创建用于存储日志的日志组和日志流：

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

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

8. 在日志组上启用不记名令牌身份验证：

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

验证配置：

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

包含的权限：自动创建的 IAM 用户将拥有以下权限：

- `logs:PutLogEvents`— 将日志事件发送到 CloudWatch 日志
- `logs:CallWithBearerToken`— 使用不记名令牌进行身份验证
- `kms:Describe*`,`kms:GenerateDataKey*`, `kms:Decrypt` — 访问 KMS 加密的日志组（条件仅限于日志服务）

选项 2：手动设置

如果您希望更好地控制 IAM 配置或需要自定义权限，则可以手动设置 HTTP 终端节点访问权限。

步骤 1：创建 IAM 用户

创建将用于日志提取的 IAM 用户：

1. 登录 AWS 管理控制台并导航到 IAM。
2. 在左侧导航窗格中，选择 用户。
3. 选择创建用户。
4. 输入用户名（例如，`cloudwatch-logs-hlc-user`）。
5. 选择下一步。
6. 附上以下 IAM 策略之一：

   选项 A：使用托管策略（推荐）

   附加CloudWatchLogsAPIKey访问管理策略。

   选项 B：创建自定义策略

   创建并附加以下 IAM 策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LogsAPIs",
            "Effect": "Allow",
            "Action": [
                "logs:CallWithBearerToken",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Sid": "KMSAPIs",
            "Effect": "Allow",
            "Action": [
                "kms:Describe*",
                "kms:GenerateDataKey*",
                "kms:Decrypt"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "kms:ViaService": [
                        "logs.*.amazonaws.com"
                    ]
                }
            },
            "Resource": "arn:aws:kms:*:*:key/*"
        }
    ]
}
```

7. 选择 “下一步”，然后选择 “创建用户”。

> **Note**
>
> 如果您计划向 KMS 加密的日志组发送日志，则需要 KMS 权限。该条件限制 KMS 只能访问通过 CloudWatch 日志服务使用的密钥。

步骤 2：生成特定于服务的凭证（API 密钥）

使用 API 生成 CloudWatch 日志 AP CreateServiceSpecificCredentialI 密钥。您也可以使用 create-service-specific-credentialCLI 命令。对于凭证期限，您可以指定一个介于 1-36600 天之间的值。如果未指定凭证期限，则 API 密钥将不会过期。

要生成有效期为 30 天的 API 密钥，请执行以下操作：

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

响应是一个ServiceSpecificCredential对象。`ServiceCredentialSecret`值是您的 CloudWatch 日志 API 密钥（不记名令牌）。

> **Important**
>
> 安全地存储 `ServiceCredentialSecret` 值，因为您以后无法检索它。如果您丢失了密钥，则需要生成一个新的 API 密钥。

步骤 3：创建日志组和日志流

创建用于存储日志的日志组和日志流：

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

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

步骤 4：启用不记名令牌身份验证

在日志组上启用不记名令牌身份验证：

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

验证配置：

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

控制生成和使用 CloudWatch 日志 API 密钥的权限

CloudWatch 日志 API 密钥的生成和使用由 CloudWatch 日志和 IAM 服务中的操作和条件密钥控制。

控制日 CloudWatch 志 API 密钥的生成

i am: CreateServiceSpecificCredential 操作控制特定于服务的密钥（例如 CloudWatch 日志 API 密钥）的生成。您可以将此操作作为一种资源，将操作范围限定为 IAM 用户，以便限制可以为其生成密钥的用户。

您可以使用以下条件密钥对 `iam:CreateServiceSpecificCredential` 操作的权限施加条件：

- iam: ServiceSpecificCredentialAgeDays — 允许您在条件中指定密钥的到期时间（以天为单位）。例如，您可以使用此条件密钥仅允许创建将在 90 天内过期的 API 密钥。
- iam: ServiceSpecificCredentialServiceName — 允许您在条件中指定服务的名称。例如，您可以使用此条件密钥仅允许为 CloudWatch 日志创建 API 密钥，而不允许为其他服务创建 API 密钥。

控制日 CloudWatch 志 API 密钥的使用

该`logs:CallWithBearerToken`操作控制 CloudWatch 日志 API 密钥的使用。要防止身份使用 L CloudWatch ogs API 密钥，请向与该密钥关联的 IAM 用户附加拒绝该`logs:CallWithBearerToken`操作的策略。

策略示例

阻止身份生成和使用 CloudWatch 日志 API 密钥

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

> **Warning**
>
> 此策略将禁止为所有支持创建特定 AWS 服务凭证的服务创建凭证。有关更多信息，请参阅 IAM 用户的服务特定凭证。

阻止身份使用日 CloudWatch 志 API 密钥

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

```
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "logs:CallWithBearerToken",
            "Resource": "*"
        }
    ]
}
```

仅当 CloudWatch 日志密钥在 90 天内过期时，才允许创建这些密钥

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

轮换 API 密钥

定期轮换 API 密钥可降低未经授权访问的风险。我们建议制定与贵组织安全政策相一致的轮换时间表。

轮换过程

要在不中断日志传输的情况下轮换 API 密钥，请按照以下步骤操作：

1. 为 IAM 用户创建新（辅助）凭证：

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

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

2. （可选）存储新凭证， AWS Secrets Manager 以便安全检索和自动轮换。
3. 将新证书导入供应商门户或更新您的应用程序配置以使用新的 API 密钥。
4. 将原始凭证设置为非活动状态：

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

5. 监控中日志组的`IncomingBytes`指标，确认日志传输不会受到影响 CloudWatch。有关更多信息，请参阅使用 CloudWatch 指标进行监控。
6. 使用新密钥确认成功交付后，删除之前的凭证：

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

监控密钥过期

要检查现有 API 密钥的创建日期和状态，请使用以下list-service-specific-credentials命令：

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

每个凭据`Status`的响应都包括`CreateDate`和。使用此信息来识别即将到期或有效时间超过轮换策略允许时间的密钥。

对泄露的 API 密钥作出响应

如果您怀疑 API 密钥已被泄露，请立即采取以下步骤：

1. 立即停用密钥以防止进一步未经授权的使用：

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

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

2. 查看 CloudTrail 日志以确定未经授权访问的范围。the section called “使用记录 API 密钥的使用情况 CloudTrail”有关如何启用 API 密钥使用情况审计，请参阅。
3. 按照中所述@@ 的轮换过程创建替换密钥the section called “轮换过程”。
4. 替换完毕后，请删除泄露的密钥：

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

5. 如果您@@ 在调查时需要立即阻止 IAM 用户的所有持有者令牌访问权限，请附上拒绝策略：

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

> **Note**
>
> 要通过 API 执行这些操作，您必须使用 AWS 凭据进行身份验证，而不是使用 CloudWatch 日志 API 密钥进行身份验证。

您还可以使用以下 IAM API 操作来管理泄露的密钥：

- ResetServiceSpecificCredential— 在不删除凭据的情况下重置密钥以生成新密码。密钥必须未过期。

API 密钥的安全最佳实践

请遵循以下最佳做法来保护您的 CloudWatch 日志 API 密钥：

- 切勿在源代码中嵌入 API 密钥。请勿在应用程序代码中对 API 密钥进行硬编码，也不要将其提交到版本控制系统。如果密钥意外被提交到公共存储库，则 AWS 自动扫描可能会对其进行标记，因此您应立即轮换密钥。
- 使用密钥管理器。将 API 密钥存储在AWS Secrets Manager或等效的机密管理解决方案中。这可以实现集中访问控制、审核日志和自动轮换。
- 为所有密钥设置过期时间。创建 API 密钥时，请务必指定一个`--credential-age-days`值。要在整个组织中强制使用最长密钥生命周期，请使用 `iam:ServiceSpecificCredentialAgeDays` IAM 条件密钥。有关示例，请参阅the section called “仅当 CloudWatch 日志密钥在 90 天内过期时，才允许创建这些密钥”。
- 应用最低权限权限。将 IAM 用户的权限范围仅限于所需的日志组和操作。使用托管CloudWatchLogsAPIKey访问策略作为起点，并根据需要进一步限制。
- 启用 CloudTrail 日志记录。通过为启用 CloudTrail 数据事件来审计 API 密钥使用情况`AWS::Logs::LogGroupAuthorization`。请参阅the section called “使用记录 API 密钥的使用情况 CloudTrail”。
- 使用 IAM 访问分析器进行监控。使用 IAM Access Analyzer 来识别与您的 API 密钥 IAM 用户关联的未使用证书和过于宽松的策略。
- 定期轮换密钥。制定轮换时间表并按照中所述的流程进行操作the section called “轮换 API 密钥”。

使用记录 API 密钥的使用情况 CloudTrail

您可以使用 AWS CloudTrail 记录日 CloudWatch 志 API 密钥使用情况的数据事件。 CloudWatch 日志会发出`CallWithBearerToken`呼叫`AWS::Logs::LogGroupAuthorization`的数据事件，使您能够审计 API 密钥何时以及如何用于发送日志。

要启用 CloudTrail 日志记录 API 密钥使用情况，请执行以下操作： CloudWatch

> **Note**
>
> 您为跟踪指定的 S3 存储桶必须具有允许 CloudTrail 向其写入日志文件的存储桶策略。有关更多信息，请参阅 Amazon S3 存储桶政策 CloudTrail。

1. 创建跟踪：

```
aws cloudtrail create-trail \
```

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

2. 配置高级事件选择器以捕获 CloudWatch 日志日志组授权事件：

```
aws cloudtrail put-event-selectors \
    --region us-east-1 \
    --trail-name cloudwatch-logs-api-key-audit \
    --advanced-event-selectors '[{
        "Name": "CloudWatch Logs API key authorization events",
        "FieldSelectors": [
            { "Field": "eventCategory", "Equals": ["Data"] },
            { "Field": "resources.type", "Equals":
 ["AWS::Logs::LogGroupAuthorization"] }
        ]
    }]'
```

3. 开始跟踪记录：

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

使用 OTLP 端点发送OpenTelemetry 日志（ 日志 ）

OpenTelemetry 日志端点 (/v1/logs) 接受 JSON 或 Protobuf 编码的 OpenTelemetry 协议 (OTLP) 日志数据。有关 OTLP 端点的详细信息，包括配置和使用情况，请参阅使用发送指标和 CloudWatch 跟 OpenTelemetry踪。

如果您使用的是不记名令牌身份验证，请先完成中的设置步骤，the section called “不记名令牌认证”然后再继续。

请求格式

- 方法：POST
- 内容类型：或 application/json application/x-protobuf
- 日志组：仅限x-aws-log-group标题（不支持查询参数）
- 日志流：x-aws-log-stream标头

示例请求

```
curl -X POST "https://logs.<region>.amazonaws.com/v1/logs" \
  -H "Authorization: Bearer ACWL<token>" \
  -H "Content-Type: application/json" \
  -H "x-aws-log-group: MyLogGroup" \
  -H "x-aws-log-stream: MyLogStream" \
  -d '{
  "resourceLogs": [
    {
      "resource": {
        "attributes": [
          {
            "key": "service.name",
            "value": { "stringValue": "my-service" }
          }
        ]
      },
      "scopeLogs": [
        {
          "scope": {
            "name": "my-library",
            "version": "1.0.0"
          },
          "logRecords": [
            {
              "timeUnixNano": "1741900000000000000",
              "severityNumber": 9,
              "severityText": "INFO",
              "body": {
                "stringValue": "User logged in successfully"
              },
              "attributes": [
                {
                  "key": "user.id",
                  "value": { "stringValue": "12345" }
                }
              ]
            }
          ]
        }
      ]
    }
```

```
  ]
}'
```

响应

成功（接受所有活动）：

```
HTTP 200 OK
{}
```

部分成功（某些事件被拒绝）：

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

当请求 Content-Type 为时`application/x-protobuf`，响应将作为具有相同字段的序列化 `ExportLogsServiceResponse` protobuf 消息返回。

OTLP 特有的行为

以下行为特定于 OTLP 端点，不存在于其他 HTTP 摄取端点上：

- Retry-After 标头 — 包含在 503 和 429 响应中，用于指示客户端何时应重试。

使用 HLC 端点发送日志（HLC 日志）

HLC 日志端点 (`/services/collector/event`) 基于 HTTP 日志收集器 (HLC) 格式。

如果您使用的是不记名令牌身份验证，请先完成中的设置步骤，the section called “不记名令牌认证”然后再继续。

输入模式

每个事件都是一个带有必填`"event"`字段的 JSON 对象。可选的元数据字段：`"time"`、`"host"`、`"source"`、`"sourcetype"`、、`"index"`。

单项赛事：

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

JSON 事件数组：

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

连接/批处理事件（没有数组包装器）：

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

事件字段（必填）

该"event"字段是必填字段。它的值可以是任何 JSON 类型：

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

不带"event"字段的对象会被静默跳过：

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

时间字段（可选）

该"time"字段以纪元秒（不是毫秒）为单位，可选小数表示亚秒精度。

格式	示例	解释为
浮点型	"time":1486683865.500	1486683865500 毫秒
整数	"time":1486683865	1486683865000 毫秒

格式	示例	解释为
字符串（浮点数）	`"time":"1486683865.500"`	1486683865500 毫秒
字符串（整数）	`"time":"1486683865"`	1486683865000 毫秒
缺失	（无时间字段）	服务器当前时间
无效	`"time":"invalid"`	服务器当前时间

Content-Type

只`application/json`被接受。

接受的 JSON 值类型

顶级类型	行为
带的对象 `"event"`	已接受
没有的物体 `"event"`	Skipped
对象数组	每个元素都单独处理
连接的对象	每个对象都单独处理
基元（字符串、数字、布尔值、空）	Skipped

终端节点格式

HLC 端点 URL 遵循以下格式：

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

必填参数：

- `<region>`— AWS 区域（例如，`us-east-1`、`eu-west-1`）

- logGroup— URL 编码的日志组名称
- logStream— URL 编码的日志流名称

可选参数：

您可以选择通过添加以下查询参数将日志事件与Service实体相关联。由于通过 HLC 端点发送的日志是自定义遥测数据，因此它们不会自动与实体关联。通过提供这些参数， CloudWatch Logs 会创建一个KeyAttributes.Type设置为Service的实体，并将其与您的日志事件相关联。这使中与 Explore 相关的功能可以 CloudWatch 将这些日志与来自同一服务的其他遥测（指标、跟踪和日志）关联起来，从而更轻松地对不同信号类型的应用程序进行故障排除和监控。有关实体和相关遥测的更多信息，请参阅向自定义遥测添加相关信息。

- entityName— 要与日志事件关联的服务实体的名称。此值存储为实体KeyAttributes.Name（例如，my-application或api.myservice.com）。
- entityEnvironment— 托管服务的环境或服务所属的环境。此值存储为实体KeyAttributes.Environment（例如productionec2:default、或eks:my-cluster/default）。

请求格式

使用 HTTP POST 发送带有以下标头和正文的日志：

标题：

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

正文格式：

请求正文应采用 JSON 格式，并包含一系列事件：

```
{
    "event": [
        {
            "time": 1730141374.001,
            "event": "Application started successfully",
            "host": "web-server-1",
            "source": "application.log",
```

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

字段描述：

- time— Unix 纪元时间戳（以秒为单位），可选十进制表示亚秒精度（可选）
- event— 日志消息或事件数据（必填）
- host— 源主机名或标识符（可选）
- source— 日志源标识符（可选）

可以根据需要添加其他自定义字段。

示例请求

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

最佳实践

对事件进行批处理

为了获得更好的性能和效率：

- 尽可能在单个请求中批量处理多个事件
- 建议的批处理大小：每个请求 10—100 个事件
- 最大请求大小：1 MB

错误处理

在应用程序中实现正确的错误处理。常见的 HTTP 状态码：

- `200 OK`— 已成功提取日志
- `400 Bad Request`— 请求格式或参数无效
- `401 Unauthorized`— 不记名令牌无效或已过期
- `403 Forbidden`— 权限不足
- `404 Not Found`— 日志组或直播不存在
- `429 Too Many Requests`— 超出速率限制
- `500 Internal Server Error`— 服务错误（使用指数退避重试）

限制

- 最大事件大小：每个事件 256 KB
- 最大请求大小：1 MB
- 每个请求的最大事件数：10,000
- 日志组名称必须遵循 CloudWatch 日志命名约定
- 如果使用不记名令牌身份验证，则必须在日志组上启用持有者令牌身份验证。

使用 NDJSON 端点发送日志（ND-JSON 日志）

ND-JSON 日志端点 (`/ingest/bulk`) 接受 NDJSON（换行符分隔 JSON）格式的日志。每行只包含一个 JSON 值，由换行符分隔。

如果您使用的是不记名令牌身份验证，请先完成中的设置步骤，the section called “不记名令牌认证”然后再继续。

请求格式

每行发送一个 JSON 值，以 `\n` (LF) 或 `\r\n` (CRLF) 分隔。空行会被静默忽略。

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

application/json和application/x-ndjson都被接受为内容类型。

接受的 JSON 值类型

根据 NDJSON 规范 (RFC 8259)，每行都接受任何有效的 JSON 值。

JSON 对象（最常见）：

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

JSON 数组（扁平化为单个事件）：

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

此单行生成 2 个事件。每个数组元素都变成一个单独的日志事件。

原始值：

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

每个基元都变成自己的事件，并带有服务器的当前时间戳。

混合类型：

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

所有 4 行都被接受为有效事件。

行内容	行为
JSON 对象	已接受，如果存在则提取时间戳

行内容	行为
JSON 数组	扁平化 — 每个元素都变成一个单独的事件
空数组 `[]`	已接受，生成 0 个事件
json 字符串	已接受为活动消息
JSON 编号	已接受为活动消息
json 布尔值	已接受为活动消息
JSON 空	已接受为活动消息
JSON 无效	已跳过（已计数，处理仍在继续）
空行	已忽略（未计为已跳过）

时间戳字段

该`"timestamp"`字段以纪元毫秒（不是秒）为单位。

格式	示例	解释为
数字（毫秒）	`"timestamp":1771007942000`	1771007942000 ms
缺失	（没有时间戳字段）	服务器当前时间
非数字	`"timestamp":"invalid"`	服务器当前时间
非对象线	`"hello"`, `42`, `true`	服务器当前时间

无效的行

无效 JSON 的行会被静默跳过并计数。下一行继续处理。

```
{"message":"valid event"}
```

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

结果：摄取 2 个事件，跳过 1 个事件。返回 HTTP 200。

如果所有行都无效，则返回HTTP 400"All events were invalid"。

示例请求

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

响应

成功（接受所有活动）：

```
HTTP 200 OK
{}
```

部分成功（某些事件被拒绝）：

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

该rejectedLogRecords字段是被拒绝的事件的总数。该errorMessage字段包含按拒绝原因划分的 JSON 编码细分：

- tooOldLogEventCount— 时间戳早于保留期的事件
- tooNewLogEventCount— 将来时间戳过远的事件
- expiredLogEventCount— 在处理过程中过期的事件

最佳实践

对事件进行批处理

为了提高性能和效率：

- 尽可能在单个请求中批量处理多个事件
- 建议的批处理大小：每个请求 10—100 个事件
- 最大请求大小：1 MB

错误处理

在应用程序中实现正确的错误处理。常见的 HTTP 状态码：

- `200 OK`— 成功提取日志
- `400 Bad Request`— 请求格式或参数无效
- `401 Unauthorized`— 不记名令牌无效或已过期
- `403 Forbidden`— 权限不足
- `404 Not Found`— 日志组或直播不存在
- `429 Too Many Requests`— 超出速率限制
- `500 Internal Server Error`— 服务错误（使用指数退避重试）

限制

- 最大事件大小：每个事件 256 KB
- 最大请求大小：1 MB
- 每个请求的最大事件数：10,000
- 日志组名称必须遵循 CloudWatch 日志命名约定
- 如果使用不记名令牌身份验证，则必须在日志组上启用持有者令牌身份验证。

使用结构化 JSON 端点发送日志（结构化 JSON 日志）

结构化 JSON 日志端点 (`/ingest/json`) 接受标准 JSON，可以是单个 JSON 对象，也可以是对象的 JSON 数组。此端点专为结构化日志数据而设计，其中每个事件都是一个 JSON 对象。

如果您使用的是不记名令牌身份验证，请先完成中的设置步骤，the section called “不记名令牌认证”然后再继续。

请求格式

仅application/json被接受为内容类型。

单个 JSON 对象：

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

JSON 对象数组：

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

接受的 JSON 值类型

此端点非常严格 — 只接受 JSON 对象作为事件。

Input	行为
单个 JSON 对象	作为一个活动被接受
JSON 对象数组	每个对象都变成一个单独的事件
空数组 []	已接受，生成 0 个事件
数组中的非对象（字符串、数字等）	Skipped
顶级原语 ("hello",42)	Skipped
连接的对象 {...}{...}	仅解析第一个对象

示例 — 混合类型的数组：

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

结果：提取了 2 个事件（对象），跳过 2 个事件（字符串和数字）。

时间戳字段

该"timestamp"字段以纪元毫秒为单位，与 NDJSON 端点相同。

格式	示例	解释为
数字（毫秒）	"timestamp":1771007942000	1771007942000 ms
缺失	（没有时间戳字段）	服务器当前时间
非数字	"timestamp":"invalid"	服务器当前时间

示例请求

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

响应

成功（接受所有活动）：

```
HTTP 200 OK
{}
```

部分成功（某些事件被拒绝）：

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

该rejectedLogRecords字段是被拒绝的事件的总数。该errorMessage字段包含按拒绝原因划分的 JSON 编码细分：

- tooOldLogEventCount— 时间戳早于保留期的事件
- tooNewLogEventCount— 将来时间戳过远的事件
- expiredLogEventCount— 在处理过程中过期的事件

最佳实践

对事件进行批处理

为了获得更好的性能和效率：

- 尽可能在单个请求中批量处理多个事件
- 建议的批处理大小：每个请求 10—100 个事件
- 最大请求大小：1 MB

错误处理

在应用程序中实现正确的错误处理。常见的 HTTP 状态码：

- 200 OK— 已成功提取日志
- 400 Bad Request— 无效的请求格式或参数
- 401 Unauthorized— 不记名令牌无效或已过期
- 403 Forbidden— 权限不足
- 404 Not Found— 日志组或直播不存在
- 429 Too Many Requests— 超出速率限制
- 500 Internal Server Error— 服务错误（使用指数退避重试）

限制

- 最大事件大小：每个事件 256 KB
- 最大请求大小：1 MB
- 每个请求的最大事件数：10,000
- 日志组名称必须遵循 CloudWatch 日志命名约定
- 如果使用不记名令牌身份验证，则必须在日志组上启用持有者令牌身份验证。

HTTP 摄取端点的比较

功能	HLC 日志	ND-JSON 日志	结构 JSON 日志	OpenTelemetry 日志
路径	`/services/collector/event`	`/ingest/bulk`	`/ingest/json`	`/v1/logs`
Content-Type	`application/json`	`application/json` 或 `application/x-ndjson`	`application/json`	`application/json` 或 `application/x-protobuf`
时间戳字段	`"time"`（秒）	`"timestamp"`（毫秒）	`"timestamp"`（毫秒）	`"timeUnixNano"`（纳秒）
必填字段	`"event"`	无	无	OTLP 结构 () `"resourceLogs"`
部分成功响应	否	是	是	是
查询参数支持	支持	是	是	否（仅限标题）

功能	HLC 日志	ND-JSON 日志	结构 JSON 日志	OpenTelemetry 日志
实体元数据	支持	是	是	否
接受基元	否	是	否	否
基于行的解析	否	是	否	否
Protobuf 支持	否	否	否	是
重试后标题	否	否	否	是

选择终端节点

- 使用 HLC 格式？ 使用 HLC 日志。您现有的 HLC 负载只需最少的更改即可运行。
- 直播 line-by-line日志？ 使用 ND-JSON 日志。最适合每行发出一个事件的日志管道。最灵活 — 接受任何 JSON 值类型。
- 正在发送结构化的 JSON 有效负载？ 使用结构化 JSON 日志。最适合生成格式良好的 JSON 对象或数组的应用程序。
- 已经在使用了 OpenTelemetry？ 使用 OpenTelemetry 日志。接受 OTLP JSON 或 Protobuf 格式，并支持带有重试语义的部分成功响应。

系统日志摄取

Amazon Logs 托管系统 CloudWatch 日志摄取允许您将系统日志消息（RFC 5424、RFC 3164 和思科 FTD/ASA）直接从防火墙、路由器、交换机和 Linux 服务器发送到日志中，无需安装或管理任何代理。 CloudWatch 您的系统日志源通过 TCP、TCP+TLS 或 UDP 向您账户中的 VPC 终端节点发送消息。流量通过隧道传输 AWS PrivateLink 到 L CloudWatch ogs syslog 服务，该服务自动解析传入消息并提取设施、严重性、主机名和应用程序名称等结构化字段，无需自定义解析管道。然后，您可以使用 CloudWatch 日志分析查询这些字段，以调查安全事件或解决连接问题。这可以帮助您集中基础架构日志可见性，简化操作工作流程，并减少在分布式环境中部署和维护日志收集代理的开销。

如果您的系统日志源已经在您的 Amazon VPC 中，它们可以直接向 VPC 终端节点发送消息。如果您的源位于外部 AWS（本地数据中心、分支机构或托管设施），则它们可以通过您的 VPN 或 Direct Connect 连接到达 VPC 终端节点。

支持的协议和端口

协议	端口：	注意
TCP + TLS	6514	传输中已加密。推荐用于合规性要求。
TCP 纯文本	1514	纯文本 AWS PrivateLink （网络隔离）。
UDP	514	Best-effort 交货。

端口 6514 上的 TLS 使用亚马逊信任服务颁发的 AWS托管证书在网络负载均衡器上终止。您的 syslog 客户端无需任何特殊配置即可自动信任此证书。

> **Note**
>
> UDP 是一种尽力而为的协议。由于网络状况，消息可能会丢失。使用 TCP 实现可靠传输。

支持的系统日志格式

CloudWatch 日志会自动检测和解析以下 syslog 格式：

- RFC 5424（较新的格式）— 包括结构化数据、ISO 8601 时间戳以及明确的应用程序名称和进程 ID 字段。
- RFC 3164（BSD 系统日志，传统格式）— 包括 BSD-style 时间戳和标签字段。仍被防火墙、路由器和交换机等网络设备广泛使用。
- 思科 FTD/ASA — 思科 Firepower 威胁防御 (FTD) 和自适应安全设备 (ASA) 设备使用的系统日志格式。消息由消息正文中的`%FTD-`或`%ASA-`标签标识。

消息以其原始原始格式存储在您的日志组中。 CloudWatch 日志会自动从每种格式中提取结构化字段，您可以使用 L CloudWatch ogs Insights 进行查询。

RFC 5424 提取的字段

字段	说明
`facility`	日志类别名称（例如，`kern`、`auth`、`local0`）。
`facilityCode`	数字设施代码 (0—23)。
`severity`	严重性级别名称（例如，`emerg`、`err`、`info`）。
`severityCode`	数字严重性代码 (0—7)。
`timestamp`	ISO 8601 格式的消息时间戳。
`hostname`	源设备主机名。
`appName`	应用程序名称。
`procId`	进程 ID。
`msgId`	消息标识符。
`structuredData`	RFC 5424 结构化数据元素（键值元数据）。
`message`	邮件正文。

RFC 3164 提取的字段

字段	说明
facility	日志类别名称。
facilityCode	数字设施代码 (0—23)。
severity	严重性级别名称。
severityCode	数字严重性代码 (0—7)。
timestamp	消息时间戳（从 BSD 格式转换为 ISO 8601）。
hostname	源设备主机名。
appName	应用程序名称（从 TAG 字段提取）。
procId	进程 ID（从 TAG 字段中提取，如果存在）。
message	邮件正文。

思科 FTD/ASA 提取的字段

字段	说明
device	设备类型（FTDASA、或FMC-AUDIT-LOG ）。
timestamp	消息时间戳（RFC 3164 或 RFC 5424 格式，取决于设备配置）。
deviceId	设备主机名（在设备上配置device-id 登录时出现）。
severity	严重性级别名称（例如，informational 、warning、critical）。
severityLevel	数字严重性级别 (0—7)。
messageId	思科数字消息标识符（例如，106023，302013）。
subsystem	子系统名称（适用于某些消息类型）。

字段	说明
message	消息正文（纯文本），或正文使用思科结构化格式时的单个键值字段。

消息传输

与服务器为每个请求返回状态码的提 HTTP-based 取不同，syslog 不向发送者提供每条消息的成功确认。该服务接收到消息后，将对其进行缓冲并传送到您的日志组，并针对暂时性错误进行重试。交付保证取决于您选择的传输协议：

- TCP（端口 6514 和 1514）— 在正常运行条件下提供可靠的传输。

  当无法交付时，该服务会重置 TCP 连接以向您的客户端发出故障信号。该连接上正在传输的消息可能会被丢弃，但是连接重置会立即提供背压，因此您的客户端可以检测到问题并在本地缓冲消息，直到问题得到解决。在容量压力下，该服务会尽早拒绝新的 TCP 连接，从而提供相同的背压信号。

  导致连接重置的条件包括：

  - VPC 终端节点策略拒绝访问
  - 已超过您账户的`PutLogEvents`配额
  - 目标日志组不存在
  - 日志组上的资源策略拒绝访问
- UDP（端口 514）— Best-effort 交付。无法传送的邮件将被丢弃，不会向发件人提供任何反馈。由于网络拥塞或容量限制，消息也可能丢失。如果可靠的交付对您的用例很重要，请使用 TCP。

要检测和响应配送问题，请监控中的`SyslogMessagesDropped`指标 CloudWatch。该`Reason`维度指出了丢弃消息的原因，因此您可以采取更正措施。有关更多信息，请参阅 the section called “监控”。

限额和限制

限制	值	注意
最大消息大小 (TCP)	64 KB	标准 syslog 消息通常远低于此限制。如果您的用例需要更大的消息，请联系客 AWS 服。

限制	值	注意
最大消息大小 (UDP)	8 KB	标准 syslog 消息通常远低于此限制。如果您的用例需要更大的消息，请联系客 AWS 服。
摄取吞吐量	与... 共享 PutLogEvents	Syslog 摄取计入您的账户PutLogEvents 配额（默认情况下，每个区域每个账户每秒 5,000 个请求）。

PutLogEvents 配额可调整。如果您的系统日志流量需要更高的吞吐量，请通过服务配额申请增加配额。

设置系统日志提取

本节将引导您完成在日志中设置 syslog 提取的步骤。 CloudWatch 您将为 syslog 服务创建 VPC 终端节点、用于接收消息的日志组、用于授权 syslog 服务的资源策略以及用于将流量从 VPC 终端节点路由到日志组的 syslog 配置。

您可以使用 AWS 管理控制台 AWS CLI、或 AWS 软件开发工具包执行所有这些步骤。以下说明提供了控制台和 AWS CLI 示例。

先决条件

用于设置 syslog 提取的 IAM 身份（用户或角色）必须具有创建 VPC 终端节点、日志组、资源策略和 syslog 配置的权限。以下示例策略显示了所需的最低权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:PutResourcePolicy",
        "logs:DeleteResourcePolicy",
        "logs:PutSyslogConfiguration",
        "logs:ListSyslogConfigurations",
        "logs:DeleteSyslogConfiguration"
      ],
```

```
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateVpc",
        "ec2:ModifyVpcAttribute",
        "ec2:CreateSubnet",
        "ec2:CreateSecurityGroup",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateVpcEndpoint",
        "ec2:ModifyVpcEndpoint",
        "ec2:DescribeVpcEndpoints"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> 如果您已经拥有 VPC、子网和安全组，则只需要执行 EC2 操作的`ec2:CreateVpcEndpointec2:ModifyVpcEndpoint`、和`ec2:DescribeVpcEndpoints`权限。

步骤 1：创建或标识 VPC

您需要一个可以从生成系统日志的设备所在的本地网络（通过 VPN 或 Direct Connect）访问的 VPC。如果您已经有一个 VPC 连接到您的数据中心，请跳过此步骤并使用现有的 VPC 和子网 ID。

Console

创建 VPC (控制台)

1. 打开位于 https://console.aws.amazon.com/vpc/ 的 Amazon VPC 控制台。
2. 在导航窗格中，选择您的 VPC，然后选择创建 VPC。
3. 对于要创建的资源，选择 仅 VPC。
4. 对于 IPv4 CIDR 块，请输入`10.0.0.0/16`（或与您的本地网络不冲突的 CIDR）。
5. 选择创建 VPC。

6. 选择新创建的 VPC，选择操作，编辑 VPC 设置。同时启用 DNS 解析和 DNS 主机名，然后选择保存。
7. 为 VPC 终端节点创建子网：
   a. 在导航窗格中，选择子网，然后选择创建子网。
   b. 对于 VPC ID，请选择您创建的 VPC。
   c. 对于可用区，请选择一个可用区。
   d. 对于 IPv4 子网 CIDR 块，请输入。`10.0.1.0/24`
   e. 选择创建子网。

AWS CLI

```
REGION=us-east-1

# Create VPC
VPC_ID=$(aws ec2 create-vpc \
  --cidr-block 10.0.0.0/16 \
  --region $REGION \
  --query 'Vpc.VpcId' --output text)

aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-support --region $REGION
aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-hostnames --region
 $REGION

# Create a subnet for the VPC endpoint
SUBNET_ID=$(aws ec2 create-subnet \
  --vpc-id $VPC_ID \
  --cidr-block 10.0.1.0/24 \
  --availability-zone ${REGION}a \
  --region $REGION \
  --query 'Subnet.SubnetId' --output text)

echo "VPC: $VPC_ID, Subnet: $SUBNET_ID"
```

> **Note**
>
> VPC 终端节点在您的子网中创建带有私有 IP 的弹性网络接口 (ENI)。您的本地设备通过您的 VPN 或 Direct Connect 连接访问此 IP。确保您的网络路由允许从您的设备流向子网 CIDR 的流量。

步骤 2：创建安全组

为 VPC 终端节点创建安全组，允许来自您的 VPC 的入站系统日志流量。这控制了哪些资源可以向端点发送 syslog。

Console

创建安全组（控制台）

1. 打开位于 https://console.aws.amazon.com/vpc/ 的 Amazon VPC 控制台。
2. 在导航窗格中，选择安全组，然后选择创建安全组。
3. 对于安全组名称，输入 `syslog-vpce-sg`。
4. 对于描述，输入 `Allow syslog traffic to VPC endpoint`。
5. 对于 VPC，请选择您在步骤 1 中创建或标识的 VPC。
6. 在入站规则部分，选择添加规则并添加以下规则：
   - 规则 1：类型 = 自定义 TCP，端口范围 =6514，来源 =10.0.0.0/16（您的 VPC CIDR）
   - 规则 2：类型 = 自定义 TCP，端口范围 =1514，来源 = 10.0.0.0/16
   - 规则 3：类型 = 自定义 UDP，端口范围 =514，来源 = 10.0.0.0/16
7. 选择创建安全组。

AWS CLI

```
VPCE_SG_ID=$(aws ec2 create-security-group \
  --group-name syslog-vpce-sg \
  --description "Allow syslog traffic to VPC endpoint" \
  --vpc-id $VPC_ID \
  --region $REGION \
  --query 'GroupId' --output text)

# Allow TCP+TLS (port 6514), TCP plaintext (port 1514), and UDP (port 514)
```

```
aws ec2 authorize-security-group-ingress --group-id $VPCE_SG_ID \
  --protocol tcp --port 6514 --cidr 10.0.0.0/16 --region $REGION
aws ec2 authorize-security-group-ingress --group-id $VPCE_SG_ID \
  --protocol tcp --port 1514 --cidr 10.0.0.0/16 --region $REGION
aws ec2 authorize-security-group-ingress --group-id $VPCE_SG_ID \
  --protocol udp --port 514 --cidr 10.0.0.0/16 --region $REGION
```

> **Tip**
>
> 如果您只打算使用一种协议（例如，端口 6514 上的 TCP+TLS），则只需在安全组中打开该端口即可。

步骤 3：创建 VPC 端点

创建指向 syslog AWS PrivateLink 服务的接口 VPC 终端节点。这为您的 VPC 提供了一个进入 CloudWatch 日志 syslog 服务的私有入口点。

> **Note**
>
> 在创建终端节点时，您可以跨不同的可用区指定多个子网 ID。终端节点在每个子网中创建一个 ENI，从而提供更高的可用性，而无需为每个可用区单独的 VPC 终端节点。子网位于多个可用区的单个 VPC 终端节点就足够了。

Console

创建 VPC 终端节点（控制台）

1. 打开位于 https://console.aws.amazon.com/vpc/ 的 Amazon VPC 控制台。
2. 在导航窗格中，选择终端节点，然后选择创建终端节点。
3. 在名称标签中，输入终端节点的名称（例如`syslog-vpce`）。
4. 对于服务类别，选择 AWS 服务。
5. 在“服务”搜索字段中，输入`syslog-logs`并选择服务`com.amazonaws.`*`Region`*`.syslog-logs`。
6. 对于 VPC，请选择您在步骤 1 中创建或标识的 VPC。
7. 在子网部分，选择一个或多个可用区，然后选择要在其中创建终端节点网络接口的子网。

8. 对于安全组，请选择您在步骤 2 (syslog-vpce-sg) 中创建的安全组。
9. （可选）如果您想限制允许通过终端节点的流量，请配置 VPC 终端节点策略。有关更多信息，请参阅 the section called “VPC 端点策略”。
10. 选择创建端点。
11. 在终端节点状态更改为 “可用” 后，选择终端节点并记下 DNS 名称值。这是您的系统日志设备将发送到的地址。

AWS CLI

```
VPCE_ID=$(aws ec2 create-vpc-endpoint \
  --vpc-id $VPC_ID \
  --service-name com.amazonaws.${REGION}.syslog-logs \
  --vpc-endpoint-type Interface \
  --subnet-ids $SUBNET_ID \
  --security-group-ids $VPCE_SG_ID \
  --region $REGION \
  --query 'VpcEndpoint.VpcEndpointId' --output text)

echo "VPC Endpoint: $VPCE_ID"
```

等待终端节点变为可用（大约 60 秒），然后检索 DNS 名称：

```
VPCE_DNS=$(aws ec2 describe-vpc-endpoints --vpc-endpoint-ids $VPCE_ID \
  --region $REGION --query 'VpcEndpoints[0].DnsEntries[0].DnsName' --output text)

echo "Endpoint DNS: $VPCE_DNS"
```

保存VPCE_DNS值 — 您将配置您的 syslog 设备以发送到此地址。

步骤 4：创建日志组

创建用于传送 syslog 消息的 Log CloudWatch s 日志组。您可以使用任何日志组名称。为了便于组织清晰起见，我们建议使用/syslog/前缀。

Console

创建日志组（控制台）

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

2. 在导航窗格的“日志管理”下，选择“日志组”。
3. 选择创建日志组。
4. 在日志组名称中，输入/syslog/my-devices。
5. （可选）根据需要配置保留设置和加密。
6. 选择创建。

AWS CLI

```
aws logs create-log-group \
  --log-group-name /syslog/my-devices \
  --region $REGION
```

您无需创建日志流。当传送第一条消息时，syslog 服务会自动创建一个名为*VPCE_ID*_Syslog_*Region*（例如vpce-0abc123def456_Syslog_us-east-1）的日志流。

步骤 5：添加资源策略

Log CloudWatch s syslog 服务使用syslog.logs.amazonaws.com服务主体写入您的日志组。您必须通过日志组的资源策略向其授予权限。该aws:SourceArn条件可确保只有来自您的特定 VPC 终端节点的流量才能写入此日志组。

```
ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)

aws logs put-resource-policy \
  --policy-name syslog-ingestion \
  --policy-document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "syslog.logs.amazonaws.com"
      },
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:'$REGION':'$ACCOUNT_ID':log-group:/syslog/my-
devices:*",
```

```
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "'$ACCOUNT_ID'"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:ec2:'$REGION':'$ACCOUNT_ID':vpc-
endpoint/'$VPCE_ID'"
        }
      }
    }
  ]
}' \
  --region $REGION
```

资源策略中的条件提供了以下保护：

- `aws:SourceAccount`— 防止混乱的副手攻击。仅接受您账户的流量。
- `aws:SourceArn`— 限定对特定 VPC 终端节点的访问权限。如果您有多个 VPC 终端节点，请在条件中将每个终端节点添加为单独的 ARN。

要允许多个 VPC 终端节点写入同一个日志组，请使用带通配符的`ArnLike`条件运算符：

```
"ArnLike": {
    "aws:SourceArn": "arn:aws:ec2:us-east-1:123456789012:vpc-endpoint/*"
}
```

步骤 6：创建系统日志配置

此步骤告知 Log CloudWatch s syslog 服务，来自您的 VPC 终端节点的流量应路由到您的日志组。如果没有此配置，来自您的终端节点的流量将被拒绝。

Console

创建 syslog 配置（控制台）

1. 打开 CloudWatch 日志控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格的 “日志管理” 下，选择 “日志组”。
3. 选择您在步骤 4 中创建的日志组（例如`/syslog/my-devices`）。
4. 在日志组详细信息中，找到 Syslog Ing estion 部分。

5. 选择配置。
6. 从 VPC 终端节点下拉列表中，选择您在步骤 3 中创建的 VPC 终端节点。
7. 选择创建。

AWS CLI

```
aws logs put-syslog-configuration \
  --log-group-identifier /syslog/my-devices \
  --vpc-endpoint-id $VPCE_ID \
  --region $REGION
```

验证配置：

```
aws logs list-syslog-configurations \
  --log-group-identifier /syslog/my-devices \
  --region $REGION
```

您的 syslog 摄取管道现已激活。发送到 VPC 终端节点的任何 syslog 消息都将传送到`/syslog/my-devices`日志组。

第 7 步：验证交付和字段提取

从任何可以到达 VPC 终端节点的 EC2 主机或设备发送测试消息，然后使用 CloudWatch Log Analytics 验证消息是否已发送以及结构化字段是否正确提取。消息通常会在 10-20 秒内出现。

发送测试消息（TCP 纯文本）：

```
echo "<134>1 $(date -u +%Y-%m-%dT%H:%M:%SZ) myhost myapp 1234 - - Hello from syslog" |
 \
  nc $VPCE_DNS 1514
```

验证交付和提取的字段：

Console

使用日志分析（控制台）验证传送

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

2. 在导航窗格的 “日志” 下，选择 “日志分析”。
3. 在日志组选择器中，选择您的日志组（例如/syslog/my-devices）。
4. 输入以下查询并选择运行查询：

```
fields @timestamp, facility, severity, hostname, appName, procId, message
| sort @timestamp desc
| limit 10
```

5. 确认您的测试消息已显示，提取的字段是否填充正确。对于上面的测试消息，您应该看到 facility severity = local0、hostname = info myhost、= myapp、appName = 和 procId = 1234。

AWS CLI

启动日志分析查询以验证交付和字段提取：

```
QUERY_ID=$(aws logs start-query \
  --log-group-name /syslog/my-devices \
  --start-time $(date -d '5 minutes ago' +%s 2>/dev/null || echo $(date -v-5M +%s))
 \
  --end-time $(date +%s) \
  --query-string 'fields @timestamp, facility, severity, hostname, appName, procId,
 message | sort @timestamp desc | limit 10' \
  --region $REGION \
  --query 'queryId' --output text)

# Wait a few seconds for the query to complete, then retrieve results
aws logs get-query-results \
  --query-id $QUERY_ID \
  --region $REGION
```

确认您的测试消息已显示，提取的字段是否填充正确。对于上面的测试消息，您应该看到 facility severity = local0、hostname = info myhost、= myapp、appName = 和 procId = 1234。

配置系统日志设备

设置系统日志提取后，将系统日志源配置为向 VPC 终端节点 DNS 名称发送消息。您的设备通过您的 VPN 或与 VPC 的 Direct Connect 连接到达终端节点。

> **Note**
>
> 以下配置作为示例提供。有关完整的配置选项和最佳实践，请参阅 syslog 实现的官方文档（rsyslog、syslog-ng 或您的设备供应商）。

在以下示例中，*VPCE_DNS*替换为设置过程中的 VPC 终端节点 DNS 名称（例如`vpce-0abc123def456.syslog-logs.us-east-1.vpce.amazonaws.com`）。

rsyslog

rsyslog 是大多数 Linux 发行版上的默认 syslog 守护程序。将以下配置添加到`/etc/rsyslog.d/50-cwl-syslog.conf`。

TCP + TLS：

```
# Forward all logs to CloudWatch Logs via TCP+TLS
action(type="omfwd"
  target="VPCE_DNS"
  port="6514"
  protocol="tcp"
  StreamDriver="gtls"
  StreamDriverMode="1"
  StreamDriverAuthMode="x509/certvalid"
  template="RSYSLOG_SyslogProtocol23Format"
  queue.type="LinkedList"
  queue.filename="cwl_fwd"
  queue.saveOnShutdown="on"
  action.resumeRetryCount="-1")
```

TCP 纯文本：

```
# Forward all logs to CloudWatch Logs via plaintext TCP
action(type="omfwd"
  target="VPCE_DNS"
  port="1514"
  protocol="tcp"
  template="RSYSLOG_SyslogProtocol23Format"
  queue.type="LinkedList"
  queue.filename="cwl_fwd"
  queue.saveOnShutdown="on"
```

```
  action.resumeRetryCount="-1")
```

UDP：

```
# Forward all logs to CloudWatch Logs via UDP
action(type="omfwd"
  target="VPCE_DNS"
  port="514"
  protocol="udp"
  template="RSYSLOG_SyslogProtocol23Format")
```

添加配置后，重新启动 rsyslog：

```
sudo systemctl restart rsyslog
```

> **Note**
>
> queue.type="LinkedList"和指queue.saveOnShutdown="on"令启用磁盘辅助队列。如果与 VPC 终端节点的连接暂时中断，rsyslog 会在本地缓冲消息并重试传送。

按设施或严重程度筛选：

要仅发送特定消息，请在操作前添加过滤器：

```
# Send only auth messages via TCP+TLS
auth,authpriv.* action(type="omfwd"
  target="VPCE_DNS"
  port="6514"
  protocol="tcp"
  StreamDriver="gtls"
  StreamDriverMode="1"
  StreamDriverAuthMode="x509/certvalid"
  template="RSYSLOG_SyslogProtocol23Format")
```

syslog-ng

syslog-ng 通常用于网络设备和企业 Linux 系统。将以下配置添加到/etc/syslog-ng/conf.d/cwl-syslog.conf。

TCP + TLS：

```
destination d_cwl_syslog {
  network("VPCE_DNS"
    port(6514)
    transport("tls")
    tls(
      peer-verify(required-trusted)
      ca-dir("/etc/ssl/certs")
    )
  );
};

log {
  source(s_sys);
  destination(d_cwl_syslog);
};
```

TCP 纯文本：

```
destination d_cwl_syslog {
  network("VPCE_DNS"
    port(1514)
    transport("tcp")
  );
};

log {
  source(s_sys);
  destination(d_cwl_syslog);
};
```

UDP：

```
destination d_cwl_syslog {
  network("VPCE_DNS"
    port(514)
    transport("udp")
  );
};

log {
  source(s_sys);
  destination(d_cwl_syslog);
```

```
};
```

添加配置后，重新启动 syslog-ng：

```
sudo systemctl restart syslog-ng
```

快速验证

在任何可以到达 VPC 终端节点的主机上使用以下命令来验证连通性。

TCP 明文（端口 1514）：

```
echo "<134>1 $(date -u +%Y-%m-%dT%H:%M:%SZ) myhost myapp 1234 - - Hello from syslog" |
 \
  nc VPCE_DNS 1514
```

TCP + TLS（端口 6514）：

```
echo "<134>1 $(date -u +%Y-%m-%dT%H:%M:%SZ) myhost myapp 1234 - - Hello TLS syslog" | \
  openssl s_client -connect VPCE_DNS:6514 -quiet -no_ign_eof
```

UDP（端口 514）：

```
echo "<134>1 $(date -u +%Y-%m-%dT%H:%M:%SZ) myhost myapp 1234 - - Hello UDP syslog" | \
  nc -u -w1 VPCE_DNS 514
```

发送测试消息后，在 CloudWatch 日志中验证传送情况（通常在 10—20 秒内）：

```
aws logs filter-log-events \
  --log-group-name /syslog/my-devices \
  --start-time $(date -d '5 minutes ago' +%s000 2>/dev/null || echo $(($(date -v-5M +
%s) * 1000))) \
  --region $REGION
```

网络设备

大多数防火墙、路由器和交换机都支持通过指定目标 IP 地址和端口来配置外部 syslog 服务器。使用 VPC 终端节点 ENI 的私有 IP 地址（在 Amazon VPC 控制台的终端节点下可见）和其中一个支持的端口（6514、1514 或 514）。

有关具体的配置语法，请参阅设备的文档。要配置的关键设置是：

- 服务器地址 — VPC 终端节点 DNS 名称或 ENI 私有 IP 地址。
- 端口 — 6514 (TCP+TLS)、1514 (TCP) 或 514 (UDP)。
- 协议-TCP（推荐）或 UDP。
- 格式 — RFC 5424（首选）或 RFC 3164（旧版）。两者都受支持。

TLS 证书信任

端口 6514 上的 TLS 连接使用亚马逊信任服务颁发的 AWS托管证书。大多数操作系统和 syslog 守护程序会自动信任此证书，因为 Amazon Trust Services 根证书包含在标准的 CA 信任存储库中。

如果您的设备默认不信任该证书，请从中下载 Amazon Trust Services 根证书，https://www.amazontrust.com/repository/并将其添加到设备的 CA 信任存储库中。

监控 syslog 摄取

syslog 摄取服务将指标发布到命名空间 CloudWatch 中。`AWS/Logs`这些指标使您可以了解您的系统日志摄取管道——收到了什么、丢失了什么以及原因。

要在 CloudWatch 控制台中查看这些指标，请导航到指标 > 所有指标 > AWS/Logs 并按下面列出的指标名称进行筛选。

系统日志指标

指标	Dimensions	说明
`SyslogMessagesReceived`	`LogGroupName`	成功摄取到日志组的 syslog 消息的数量。
`SyslogMessagesDropped`	`LogGroupName`，`Reason`	无法传送的系统日志消息的数量。有关详细信息，请参阅the section called “掉线原因”。
`SyslogConnectionsRejected`	`Reason`	被拒绝的 TCP 连接的数量。
`SyslogConnectionsEstablished`	—	成功接受的 TCP 连接的数量。

指标	Dimensions	说明
SyslogConnectionsClosed	—	关闭的 TCP 连接的数量。

掉线原因

该SyslogMessagesDropped指标包括一个Reason维度，该维度表明了消息被删除的原因。

Reason	说明
MessageRateLimitExceeded	已超过您账户的PutLogEvents 配额。考虑申请增加配额。
MessageSizeExceeded	UDP 数据报超过了最大消息大小。
ServiceUnavailable	已超过内部容量或速率限制。这通常是短暂的。
ResourceNotFound	目标日志组不存在。确认该日志组未被删除。
AccessDenied	日志组上的资源策略不授予对 syslog 服务的访问权限。验证资源策略是否正确。
VpcePolicyDenied	VPC 终端节点策略拒绝了该请求。查看您的 VPC 终端节点政策。
InternalError	发生了意外的内部错误。如果这种情况仍然存在，请联系 AWS 支持部门。

连接被拒绝的原因

该SyslogConnectionsRejected指标包括一个Reason维度。

Reason	说明
VpcePolicyDenied	VPC 终端节点策略拒绝了连接。

Reason	说明
ServiceUnavailable	由于内部错误，连接中断。

建议的警报

我们建议在以下条件下创建 CloudWatch 警报，以便尽早发现问题：

警报	条件	建议采取的措施
留言被删除	SyslogMessagesDropped > 0 持续 5 分钟	调查Reason维度以确定原因。
拒绝访问	SyslogMessagesDropped 与 Reason=AccessDenied	验证日志组的资源策略配置是否正确。
日志组丢失	SyslogMessagesDropped 与 Reason=ResourceNotFound	确认该日志组存在且未被删除。
未收到任何消息	SyslogMessagesReceived = 0 持续 15 分钟（通常情况下 > 0）	验证设备是否仍在发送以及与 VPC 终端节点的网络连接完好无损。

有关创建 CloudWatch 警报的信息，请参阅《CloudWatch 用户指南》中的创建 CloudWatch 警报。

在自动仪表板上查看 syslog 指标

CloudWatch 为 CloudWatch 日志提供自动控制面板，其中包含 Syslog 摄取部分。当您的账户发布系统日志指标时，此仪表板会自动显示——无需手动设置。该仪表板包含中描述的每个指标的图表the section called “系统日志指标”。每个图表显示按相关维度（例如日志组或原因）细分的指标速率。

访问控制面板

要查看 syslog 自动控制面板，请执行以下操作：

1. 在处打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/。

2. 在导航窗格中，选择仪表板，然后选择自动仪表板。

3. 选择CloudWatch 日志控制面板。

4. 滚动到 Syslog 摄取部分。

系统日志数据的示例查询

以下 CloudWatch Logs Insights 查询演示了系统日志数据的常见分析模式。它们使用 CloudWatch 日志自动从 RFC 5424、RFC 3164 和思科消息中提取的结构化字段。 FTD/ASA 有关提取字段的完整列表，请参阅the section called “支持的系统日志格式”。

CloudWatch 控制台中也提供这些查询。在 “ CloudWatch 日志见解” 中，选择查询，然后展开 Syslog 类别。

安全和访问监控

Sudo 和权限升级事件

以下查询列出了最近在 Linux 主机上进行的 sudo 权限升级活动。它过滤sudo应用程序名称，并处理 RFC 3164 和 RFC 5424 消息。

```
filter appName="sudo"
| fields @timestamp, hostname, message
| sort @timestamp desc
| limit 50
```

SSH 身份验证失败的前 50 台主机

以下查询计算了每台主机失败的 SSH 身份验证尝试次数。它匹配包含 “密码失败” 或 “公钥失败” 的 sshd 消息。使用它来识别成为暴力攻击目标的主机。

```
filter appName="sshd" and (message like /Failed password/ or message like /Failed
 publickey/)
| stats count(*) as failedAttempts by hostname
| sort failedAttempts desc
| limit 50
```

运营可见性

一段时间内按主机排列的日志量

以下查询以 5 分钟为间隔计算每台主机的消息数。使用它来测量队列活动、识别音量峰值或检测已停止发送的主机。此查询适用于所有 syslog 格式类型。

```
stats count(*) as messages by hostname, bin(5m)
| sort messages desc
```

按设施和严重程度划分的消息量

以下查询按设施和严重性级别对消息进行计数。使用它来了解哪些子系统产生的日志流量最多，严重程度如何。

```
stats count(*) as messageCount by facility, severity
| sort messageCount desc
```

High-severity 按应用程序排列的消息

以下查询筛选出高优先级消息（SeverityCode 0—3，对应于出现、警报、暴击和错误），并按应用程序对其进行计数。使用它来确定哪些应用程序生成的错误最多。

```
filter severityCode<=3
| stats count(*) as errorCount by appName
| sort errorCount desc
```

Warning-and-above 随着时间的推移趋势

以下查询以 5 分钟为间隔对严重程度为警告或更高（严重性代码 0—4）的消息进行计数。将其与 Logs Insights 中的可视化功能配合使用，绘制一段时间内的错误峰值。

```
filter severityCode<=4
| stats count(*) as events by bin(5m)
| sort @timestamp
```

防火墙和网络设备查询

思科拒绝连接 (ACL 106023)

以下查询适用于思科 ASA 和 FTD 防火墙。它对按设备分组的 ACL-denied连接（思科消息 ID 106023）进行计数，这有助于您识别被拒绝流量最多的设备。

```
filter messageId="106023"
| stats count(*) as denies by deviceId
| sort denies desc
| limit 20
```

按消息 ID 划分的思科 ASA/FTD 事件

以下查询适用于思科 FTD 和 ASA 防火墙。它对按思科消息 ID 和严重性分组的事件进行计数，以帮助您确定防火墙队列中最常见的事件类型。

```
filter ispresent(messageId)
| stats count(*) as eventCount by messageId, severity
| sort eventCount desc
```

防火墙按源 IP 丢弃了数据包

以下查询解析防火墙丢弃消息（例如，来自 iptables 或 nftables），以按源 IP 地址统计丢弃的数据包。使用它来识别封锁流量的主要来源。

```
filter message like /DROP/
| parse message /SRC=(?<srcAddr>\S+)/
| stats count(*) as drops by srcAddr
| sort drops desc
| limit 20
```

系统日志的 VPC 终端节点策略

您可以将 VPC 终端节点策略附加到您的 syslog VPC 终端节点，以控制允许哪些流量通过该终端节点。除了安全组和资源策略之外，这还提供了额外的访问控制层。

默认情况下，VPC 终端节点具有允许所有流量的完全访问策略。您可以通过附加自定义策略来限制这一点。有关 VPC 终端节点策略的更多信息，请参阅AWS PrivateLink 指南中的使用终端节点策略控制访问。

> **Important**
>
> Syslog 流量以匿名委托人的身份到达（网络上没有 IAM 身份）。策略必须使用操作、资源或 VPC 条件键来使用"Principal": "*"和限定访问范围。 Principal-scoped 策略（例如，仅允许特定 IAM 角色的策略）将拒绝所有 syslog 流量。

支持的条件键

系统日志的 VPC 终端节点策略支持以下条件键：

条件键	说明	示例值
aws:SourceVpc	流量来源的 VPC。	vpc-111bbb22
aws:SourceVpce	流量到达的 VPC 终端节点。	vpce-0abc123def456
aws:VpcSourceIp	VPC 内发送方的私有 IP 地址。	10.0.1.50
aws:ResourceAccount	拥有目标日志组的账户。	123456789012
aws:ResourceOrgID	拥有目标日志组的账户的组织 ID。	o-aa111bb222
aws:ResourceOrgPaths	拥有目标日志组的账户的组织单位路径。	o-aa111bb222/r-ab12/ou-ab12-11111111/*

由于系统日志流量没有 IAM 委托人身份，因此不支持以下条件键：

- aws:PrincipalArn
- aws:PrincipalAccount
- aws:PrincipalOrgID
- aws:PrincipalOrgPaths
- aws:PrincipalTag/*

使用不支持的条件密钥的策略将导致隐式拒绝，从而阻止通过端点的所有系统日志流量。

限制到特定的日志组

以下策略仅允许发送到特定日志组的 syslog 流量：

```
aws ec2 modify-vpc-endpoint \
  --vpc-endpoint-id $VPCE_ID \
  --policy-document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "logs:PutLogEvents",
      "Resource": "arn:aws:logs:Region:account-id:log-group:/syslog/my-devices"
    }
  ]
}' \
  --region $REGION
```

拒绝所有流量

以下策略阻止通过端点的所有系统日志流量。这对于在不删除端点的情况下暂时停止摄取很有用：

```
aws ec2 modify-vpc-endpoint \
  --vpc-endpoint-id $VPCE_ID \
  --policy-document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "*",
      "Resource": "*"
    }
  ]
}' \
  --region $REGION
```

重置为默认策略

要恢复默认的全部允许策略，请执行以下操作：

```
aws ec2 modify-vpc-endpoint \
  --vpc-endpoint-id $VPCE_ID \
  --reset-policy \
  --region $REGION
```

策略传播

创建或修改 VPC 终端节点策略后，策略更改最多可能需要几分钟才能生效。在此传播期内，之前的策略（或默认策略）将继续适用。

如果 VPC 终端节点策略拒绝 syslog 流量，您将看到以下指标：

- 策略评估后，TCP 连接会立即关闭。
- UDP 数据报被丢弃。
- 会发出带有`Reason=VpcePolicyDenied`的`SyslogMessagesDropped`指标。
- 带的`SyslogConnectionsRejected`指标`Reason=VpcePolicyDenied`是针对 TCP 连接发出的。

对系统日志摄取进行故障排除

本节介绍了 syslog 摄取的常见问题以及如何解决这些问题。

消息未出现在日志组中

如果您发送了 syslog 消息，但这些消息未出现在您的日志组中，请按顺序检查以下内容：

1. 验证系统日志配置是否存在。

   运行`list-syslog-configurations`以确认您的 VPC 终端节点已在日志组中注册：

   ```
   aws logs list-syslog-configurations \
     --log-group-identifier /syslog/my-devices \
     --region $REGION
   ```

   如果未返回任何配置，请使用创建配置`put-syslog-configuration`。

2. 验证资源策略。

   确保日志组的资源策略授予`logs:PutLogEvents`和`syslog.logs.amazonaws.com`，并且`aws:SourceArn`条件`logs:CreateLogStream`与您的 VPC 终端节点 ARN 完全匹配：

   ```
   aws logs describe-resource-policies --region $REGION
   ```

   使用检查`SyslogMessagesDropped`指标`Reason=AccessDenied`以确认授权是否是问题所在。

3. 验证安全组。

   确保连接到 VPC 终端节点的安全组允许来自源 CIDR 的入站流量进入您正在使用的端口（6514、1514 或 514）：

   ```
   aws ec2 describe-security-groups \
     --group-ids $VPCE_SG_ID \
     --region $REGION \
     --query 'SecurityGroups[0].IpPermissions'
   ```

4. 验证网络连接。

   在同一网络上的设备或主机上，测试与端点的 TCP 连接：

   ```
   # TCP connectivity test
   nc -zv $VPCE_DNS 6514
   nc -zv $VPCE_DNS 1514
   ```

   如果连接被拒绝或超时，请验证 VPN/Direct Connect 路由以及设备是否可以到达 VPC 终端节点子网。

5. 检查 VPC 终端节点策略。

   如果您已附加自定义 VPC 终端节点策略，请确认它允许执行所需的操作。使用检查`SyslogMessagesDropped`指标`Reason=VpcePolicyDenied`。

TLS 握手失败

如果您的 syslog 客户端无法在端口 6514 上建立 TLS 连接：

- 证书信任 — 确保您设备的 CA 信任存储区包含 Amazon Trust Services 根证书。默认情况下，大多数操作系统都包含这些。https://www.amazontrust.com/repository/如果需要，可以从中下载它们。

- TLS 版本 — 该服务支持 TLS 1.2 和 TLS 1.3。确保您的系统日志客户端至少支持 TLS 1.2。
- 主机名验证 — 如果您的客户端验证服务器主机名，请确保将其配置为使用 VPC 终端节点 DNS 名称（不是 IP 地址）进行连接。

使用 OpenSSL 手动测试 TLS 连接：

```
openssl s_client -connect $VPCE_DNS:6514 -brief
```

成功连接会显示 TLS 版本和密码套件。如果失败，则错误消息会指出原因。

UDP 消息丢失

UDP 是一种尽力而为的协议。在以下情况下，某些消息丢失是预期的行为：

- 您的设备和 VPC 终端节点之间的网络拥塞。
- 消息超过了最大 UDP 数据报大小。
- 临时服务容量限制（`SyslogMessagesDropped`用用表示`Reason=ServiceUnavailable`）。

如果需要可靠的传输，请改用 TCP（端口 1514 或 6514）。TCP 提供传送确认和自动重传功能。

连接被拒绝

如果您的设备在连接到 VPC 终端节点时报告 “连接被拒绝”：

- 验证 VPC 终端节点的`available`状态为：

  ```
  aws ec2 describe-vpc-endpoints --vpc-endpoint-ids $VPCE_ID \
    --region $REGION --query 'VpcEndpoints[0].State'
  ```

- 确认您正在使用支持的端口（6514、1514 或 514）进行连接。
- 验证 VPN 或 Direct Connect 路由允许从您的设备网络到 VPC 终端节点子网的流量。
- 检查安全组是否允许您使用的端口上的入站流量。

由于速率限制而丢弃了邮件

如果`SyslogMessagesDropped`指标显示`Reason=MessageRateLimitExceeded`，则表示您的系统日志流量已超过您账户的`PutLogEvents`配额。

要解决这一问题，请执行以下操作：

1. 在 Amazon Lo CloudWatch gs 下的 S ervice Quotas 控制台中查看您当前的配额。
2. 如果您的系统日志吞吐量需要，请申请增加配额。请参阅 Service Quotas 用户指南中的申请增加配额。

将 L CloudWatch ogs 与 AWS SDK

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
适用于 C++ 的 AWS SDK	适用于 C++ 的 AWS SDK 代码示例
AWS CLI	AWS CLI 代码示例
适用于 Go 的 AWS SDK	适用于 Go 的 AWS SDK 代码示例
适用于 Java 的 AWS SDK	适用于 Java 的 AWS SDK 代码示例
适用于 JavaScript 的 AWS SDK	适用于 JavaScript 的 AWS SDK 代码示例
适用于 Kotlin 的 AWS SDK	适用于 Kotlin 的 AWS SDK 代码示例
适用于 .NET 的 AWS SDK	适用于 .NET 的 AWS SDK 代码示例
适用于 PHP 的 AWS SDK	适用于 PHP 的 AWS SDK 代码示例
AWS Tools for PowerShell	AWS Tools for PowerShell 代码示例
AWS 适用于 Python 的 SDK v4	AWS 适用于 Python 的 SDK v4 代码示例
适用于 Python (Boto3) 的 AWS SDK	适用于 Python (Boto3) 的 AWS SDK 代码示例
适用于 Ruby 的 AWS SDK	适用于 Ruby 的 AWS SDK 代码示例
适用于 Rust 的 AWS SDK	适用于 Rust 的 AWS SDK 代码示例
适用于 SAP ABAP 的 AWS SDK	适用于 SAP ABAP 的 AWS SDK 代码示例
适用于 Swift 的 AWS SDK	适用于 Swift 的 AWS SDK 代码示例

有关特定于 CloudWatch 日志的示例，请参阅使用 CloudWatch 日志的代码示例 AWS 软件开发工具包。

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。

日志管理

CloudWatch 日志提供高级日志管理功能，可帮助您更有效地组织、转换和分析日志数据。这些功能包括跨账户和跨区域数据集中、自动数据发现和架构管理、采集期间的日志转换，以及使用用于交互式日志探索的各个方面的增强分析。

主题

- 数据源发现和管理
- 数据源启用的功能
- 支持的数据来源

数据源发现和管理

CloudWatch Logs 会根据数据源和类型自动发现您的日志数据并对其进行分类，从而更轻松地大规模理解和管理您的日志。此功能为提供来源（例如 Amazon VPC 流日志和 Route 53）以及第三方安全工具提供架构发现。 AWS CloudTrail

日志管理控制台提供了按数据源和类型组织的日志的高级视图，而不仅仅是日志组。该组织可以帮助您：

- 查看按 AWS 服务、第三方来源（例如 Okta 或 CrowdStrike）和自定义来源分类的日志
- 自动了解日志数据的架构和结构
- 根据发现的架构字段创建字段索引策略
- 更高效地管理不同数据源的日志
- 按不同数据源查询日志

当您为支持的 AWS 服务启用 CloudWatch 日志记录时， CloudWatch Logs 会自动将相应的架构应用于您的日志。这个自动架构应用程序有助于保持一致性，并提供对日志结构的即时见解。

什么是 CloudWatch 日志数据源？

CloudWatch Logs Data Sources 是一项功能，它提供了一种根据生成日志的来源对日志数据进行组织和分类的新方法。虽然 CloudWatch 日志传统上使用日志组来组织日志，但数据源提供了额外的组织层，可根据日志的源服务和日志类型对日志进行分组。

数据源的工作原理

数据源提供基于服务的日志组织，并简化了整个 AWS 基础架构的发现。您可以轻松地从特定服务中找到日志并按日志类型进行筛选，而无需知道各个日志组的名称或结构。

对于第三方来源，也可以选择应用程序日志源，数据源使用 CloudWatch 管道对您的日志进行分类。将管道配置为采集和转换日志时，需要指定数据源的名称和类型。 CloudWatch 然后，日志会自动对管道处理的所有日志进行分类。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的CloudWatch 管道。

数据源使用两个关键标识符对日志进行分类：

- 数据源名称：生成日志的 AWS 服务、第三方来源或应用程序（例如 Route 53、Amazon VPC CloudTrail、Okta SSO 或 CrowdStrike Falcon）。
- 数据源类型：该服务生成的特定日志类型。

架构定义了日志数据的结构，包括存在哪些字段以及信息的组织方式。单个数据源可以生成具有不同架构和用途的多种类型的日志。例如， AWS CloudTrail 数据源有两种类型：管理事件（跟踪控制平面操作，例如创建或删除资源）和数据事件（跟踪数据平面操作，例如 S3 对象访问）。每种类型都有不同的架构，因为它们捕获的信息类型不同。

如何开始

CloudWatch 日志根据您的数据源将您的日志归类到数据源中。该方法取决于你正在处理的日志的类型：

AWS 服务 logs

来自支持的日志 AWS 服务会自动按数据源分组，无需任何配置。 CloudWatch 日志可以识别这些日志，并根据源服务应用相应的数据源名称和类型。

Third-party 日志

Third-party 日志需要管道来对数据源进行分类。将管道配置为从支持的第三方来源（例如 Microsoft Office 365、Okta 或 Palo Alto Networks）提取日志时，需要在管道配置中指定数据源名称和类型。 CrowdStrike CloudWatch 日志使用这些标识符自动对管道处理的所有日志进行分类。

Pipelines 可以选择将第三方日志转换为开放网络安全架构框架 (OCSF) 格式，用于标准化安全事件分析。启用 OCSF 转换后，将根据 OCSF 架构映射自动确定数据源的名称和类型。如果没有 OCSF 转换，则可以在工作流配置中指定数据源名称和类型。

应用程序日志

对于自定义应用程序日志，您可以使用以下方法之一按数据源对其进行分类：

- 日志组标签-使用密钥向日志组添加标签，`cw:datasource:name`并`cw:datasource:type`分别为日志组中提取的所有日志指定数据源名称和类型。标签值最多可包含 64 个字符，并且只能包含小写字母、数字和下划线。它们必须以字母或数字开头，并且不能包含双下划线 (__)。
- 管道配置-在摄取应用程序日志时，通过日志处理管道配置数据源信息。

> **Note**
>
> 数据源名称不能以 “aws” 或 “amazon” 开头，以避免与 AWS 服务日志发生冲突。

系统字段

CloudWatch 日志会自动向按数据源分类的日志中添加三个系统字段。这些字段用作默认分面：

- `@data_source_name`-包含数据源的名称，如果未确定，则为 “未知”
- `@data_source_type`-包含数据源的类型，如果未确定，则为 “未知”
- `@data_format`-表示日志数据的格式

当无法确定数据源名称或类型时，这些字段将设置为 “未知”。带有 “未知” 值的数据源在分面和控制台中 “日志管理” 下的数据源表中仍然可见，这使您可以识别未分类的日志以及它们来自哪个日志组。

该`@data_format`字段可以包含以下值之一：

- `Default`-未经修改即采集的日志。
- `Custom`-通过管道处理器处理的日志或载入带有数据源 name/type 标签的日志组的日志。
- `OCSF-<version>`-在管道中使用 OCSF（开放网络安全架构框架）处理器处理的日志。
- `AWS-OTEL-LOG-V<version>`-通过 CloudWatch OTLP 端点提取的 OpenTelemetry 日志。
- `AWS-OTEL-TRACE-V<version>`-通过 CloudWatch OTLP 端点摄取的 OpenTelemetry 跟踪。

这些系统字段使您能够根据日志的来源和格式筛选和查询日志，从而更轻松地处理来自不同来源和处理管道的日志。

访问数据源

控制台

在 CloudWatch 日志控制台中，您可以使用日志管理选项卡访问您的数据源。 CloudWatch 日志会自动按数据源和类型整合您的日志数据，持续发现新摄取的数据。在数据源列表中，您可以创建管道、定义字段索引和分面。

AWS CLI

使用以下命令列出您账户中的不同数据源和日志类型：

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

与日志组的关系

数据源补充而不是取代日志组。您的日志仍像以前一样存储在日志组中，但现在它们也会自动使用数据源信息进行标记。这种双重组织使您能够：

- 使用日志组制定精细的访问控制和保留策略
- 使用数据源进行基于服务的日志发现和分析
- 根据需要使用任一组织方法查询日志

数据源通过在整个 AWS 基础架构中提供以服务为中心的日志数据视图，可以更轻松地大规模处理日志。

数据源启用的功能

数据源通过现场发现和一致的数据结构实现高级日志处理和分析功能。

- 分@@ 面：分面是索引日志字段，无需编写查询即可提供交互式筛选和分析。 CloudWatch 日志会自动为数据源名称和类型创建分面，您可以对发现的字段创建分面策略以加快故障排除速度。Facets 在 Logs Insight CloudWatch s 中显示值分布和计数，因此通过点选探索可以轻松识别模式。
- 管道：创建适用于来自特定数据源名称和类型的所有日志的转换管道。这允许您为来自同一来源的日志定义一致的处理规则。
- 字段发现： CloudWatch 日志会根据管道处理器自动发现每个数据源名称和类型组合的字段及其数据类型。对于 AWS 托管日志，字段结构是预定义的。对于应用程序日志，我们建议保持一致的日志格式，以最大限度地兼容需要明确定义的字段结构的 Amazon S3 表等分析工具。

您可以使用 GetLogFields API 查看任何数据源的字段及其类型的完整列表：

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

这种字段发现和一致性支持高级分析和集成，因为在处理日志数据时，外部工具可以使用可预测的字段结构。

支持的数据来源

CloudWatch 日志支持各种数据源，包括 AWS 服务、第三方安全和 IT 工具以及自定义应用程序源。此页面提供了所有支持的数据源的全面目录。有关如何发现和管理数据源的信息，请参阅the section called “数据源发现和管理”。

主题

- 支持 AWS 服务 用于数据源
- 数据源支持的第三方来源
- 自定义来源

支持 AWS 服务 用于数据源

下表列出了由 L AWS 服务 ogs 自动归 CloudWatch 类为数据源的内容：

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
amazon_api_gateway	access
amazon_bedrock_agentcore	browser_usage
amazon_bedrock_agentcore	code_interpreter_application
amazon_bedrock_agentcore	code_interpreter_usage
amazon_bedrock_agentcore	gateway_application
amazon_bedrock_agentcore	identity_workload_application
amazon_bedrock_agentcore	memory_application

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
amazon_bedrock_agentcore	online_evaluation_config
amazon_bedrock_agentcore	runtime_application
amazon_bedrock_agentcore	runtime_usage
amazon_bedrock_agents	application
amazon_bedrock_agents	event
amazon_bedrock_knowledge_bases	application
amazon_cloudfront	access
amazon_cloudfront	connection
amazon_cloudwatch	rum_app_monitor
amazon_cognito	user_pool
amazon_ec2	verified_access
amazon_eks	api_server
amazon_eks	audit
amazon_eks	authenticator
amazon_eks	controller_manager
amazon_eks	scheduler
amazon_elasticache	cluster
amazon_eventbridge	eventbus_error
amazon_eventbridge	eventbus_info
amazon_eventbridge	pipes_execution

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
amazon_interactive_video_service	chat
amazon_managed_prometheus	scraper
amazon_managed_prometheus	workspace
amazon_msk	broker
amazon_msk	connect
amazon_opensearch_service	pipeline
amazon_q_business	events
amazon_q_business	sync_job
amazon_q_connect	events
amazon_route53	global_resolver_query
amazon_route53	hosted_zones
amazon_route53	profiles_resolver_query
amazon_route53	resolver_query
amazon_sagemaker	workteam_activity
amazon_ses	ingress_endpoints
amazon_ses	rule_sets
amazon_ses	traffic_policy
amazon_vpc	flow
amazon_vpc	route_server_peer
amazon_vpc_lattice	access

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
amazon_vpc_lattice	resource_access
amazon_workmail	access_control
amazon_workmail	authentication
amazon_workmail	personal_access
amazon_workmail	workmail_access
amazon_workmail	workmail_availability
aws_b2b_data_interchange	execution
aws_backup	data_access
aws_backup	hypervisor
aws_clean_rooms	analysis
aws_client_vpn	connection
aws_client_vpn	event
aws_cloudtrail	data
aws_cloudtrail	management
aws_elemental_mediapackage	egress_access
aws_elemental_mediapackage	ingress_access
aws_elemental_mediatailor	ad_decision
aws_elemental_mediatailor	manifest
aws_elemental_mediatailor	transcode
aws_entity_resolution	id_mapping_workflow

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
aws_entity_resolution	matching_workflow
aws_iot_fleetwise	error
aws_mainframe_modernization	batch_job
aws_mainframe_modernization	config
aws_mainframe_modernization	console
aws_mainframe_modernization	dataset_import
aws_network_firewall	alert
aws_network_firewall	flow
aws_network_firewall	tls
aws_nlb	access
aws_pcs	job_completion
aws_pcs	scheduler
aws_security_hub	compliance_finding
aws_security_hub	data_security_finding
aws_security_hub	detection_finding
aws_security_hub	vulnerability_finding
aws_security_hub_cspm	asff_finding
aws_shield	protection_flow
aws_step_functions	express
aws_step_functions	standard

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
aws_transfer_family	server
aws_waf	access

数据源支持的第三方来源

下表列出了通过管道摄取时被 Log CloudWatch s 自动归类为数据源的第三方来源：

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
akamai_datastream_2	base_event
akamai_datastream_2	dns_activity
akamai_datastream_2	http_activity
box	account_change
box	authentication
box	detection_finding
box	entity_management
box	file_hosting_activity
box	group_management
box	user_inventory_info
broadcom_carbonblack	authentication
broadcom_carbonblack	detection_finding
broadcom_carbonblack	file_system_activity
broadcom_carbonblack	module_activity

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
broadcom_carbonblack	network_activity
broadcom_carbonblack	process_activity
broadcom_carbonblack	registry_key_activity
broadcom_carbonblack	registry_value_activity
broadcom_carbonblack	script_activity
checkpoint_nextgenerationfirewall	data_security_finding
checkpoint_nextgenerationfirewall	detection_finding
checkpoint_nextgenerationfirewall	entity_management
checkpoint_nextgenerationfirewall	network_activity
cisco_duo	authentication
cisco_duo	base_event
cisco_duo	device_inventory_info
cisco_duo	entity_management
cisco_duo	user_inventory_info
cisco_ftd	authentication
cisco_ftd	authorize_session
cisco_ftd	detection_finding
cisco_ftd	network_activity

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
cisco_ftd	tunnel_activity
cisco_ise	account_change
cisco_ise	api_activity
cisco_ise	application_error
cisco_ise	application_lifecycle
cisco_ise	authentication
cisco_ise	entity_management
cisco_ise	network_activity
cisco_ise	tunnel_activity
cisco_ise	web_resources_activity
cisco_meraki	api_activity
cisco_meraki	detection_finding
cisco_meraki	network_activity
cisco_umbrella	data_security_finding
cisco_umbrella	dns_activity
cisco_umbrella	entity_management
cisco_umbrella	network_activity
crowdstrike_falcon	detection_finding
crowdstrike_falcon	process_activity
drupal_core	application_lifecycle

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
drupal_core	authentication
drupal_core	entity_management
drupal_core	http_activity
entrust_idaas	authentication
entrust_idaas	entity_management
f5_bigip	http_activity
f5_bigip	network_activity
github_auditlogs	account_change
github_auditlogs	api_activity
github_auditlogs	entity_management
gitlab	group_management
gitlab	software_inventory_info
gitlab	user_inventory_info
gitlab	vulnerability_finding
gitlab	web_resources_activity
google_workspace	account_change
google_workspace	api_activity
google_workspace	authentication
google_workspace	base_event
google_workspace	data_security_finding

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
google_workspace	detection_finding
google_workspace	email_activity
google_workspace	entity_management
google_workspace	file_hosting_activity
google_workspace	user_access_management
google_workspace	web_resources_activity
hashicorp_vault	api_activity
jamf_pro	authentication
jamf_pro	device_inventory_info
jamf_pro	entity_management
jamf_protect	detection_finding
jamf_protect	device_inventory_info
jamf_protect	dns_activity
jamf_protect	software_inventory_info
jamf_protect	vulnerability_finding
jumpcloud	account_change
jumpcloud	authentication
jumpcloud	datastore_activity
jumpcloud	detection_finding
jumpcloud	entity_management

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
microsoft_entraid	account_change
microsoft_entraid	authentication
microsoft_entraid	entity_management
microsoft_entraid	user_access_management
microsoft_office365	account_change
microsoft_office365	application_lifecycle
microsoft_office365	authentication
microsoft_office365	compliance_finding
microsoft_office365	detection_finding
microsoft_office365	email_activity
microsoft_office365	file_hosting_activity
microsoft_office365	group_management
microsoft_office365	incident_finding
microsoft_office365	user_access_management
microsoft_office365	vulnerability_finding
microsoft_office365	web_resources_activity
microsoft_windows	account_change
microsoft_windows	authentication
microsoft_windows	entity_management
microsoft_windows	event_log_activity

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
microsoft_windows	file_system_activity
microsoft_windows	group_management
microsoft_windows	kernel_activity
netskope_cloudexchange	account_change
netskope_cloudexchange	authentication
netskope_cloudexchange	data_security_finding
netskope_cloudexchange	detection_finding
netskope_cloudexchange	device_inventory_info
netskope_cloudexchange	entity_management
netskope_cloudexchange	file_hosting_activity
netskope_cloudexchange	network_activity
okta_auth0	api_activity
okta_auth0	authentication
okta_sso	api_activity
okta_sso	authentication
okta_sso	detection_finding
okta_sso	entity_management
onelogin_identity	account_change
onelogin_identity	authentication
onelogin_identity	entity_management

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
paloaltonetworks_nextgenerationfirewall	authentication
paloaltonetworks_nextgenerationfirewall	detection_finding
paloaltonetworks_nextgenerationfirewall	network_activity
paloaltonetworks_nextgenerationfirewall	process_activity
paloaltonetworks_prismacloud	base_event
paloaltonetworks_prismacloud	cloud_resources_inventory_info
paloaltonetworks_prismacloud	compliance_finding
paloaltonetworks_prismacloud	detection_finding
paloaltonetworks_prismacloud	device_inventory_info
paloaltonetworks_prismacloud	entity_management
paloaltonetworks_prismacloud	scan_activity
paloaltonetworks_prismacloud	vulnerability_finding
pingidentity_pingaccess	api_activity
pingidentity_pingaccess	http_activity
pingidentity_pingfederate	authentication
pingidentity_pingfederate	entity_management
pingidentity_pingone	account_change
pingidentity_pingone	authentication

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
pingidentity_pingone	entity_management
proofpoint_tap	detection_finding
proofpoint_tap	email_activity
proofpoint_tap	osint_inventory_info
qualys_csam	device_inventory_info
qualys_csam	osint_inventory_info
qualys_csam	software_inventory_info
qualys_csam	vulnerability_finding
qualys_policyaudit	compliance_finding
qualys_vmdr	account_change
qualys_vmdr	api_activity
qualys_vmdr	authentication
qualys_vmdr	device_inventory_info
qualys_vmdr	entity_management
qualys_vmdr	vulnerability_finding
sailpoint_isc	account_change
sailpoint_isc	authentication
sailpoint_isc	entity_management
sailpoint_isc	user_access_management
salesforce_elf	account_change

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
salesforce_elf	api_activity
salesforce_elf	authentication
salesforce_elf	datastore_activity
salesforce_elf	detection_finding
salesforce_elf	entity_management
salesforce_elf	file_hosting_activity
salesforce_elf	http_activity
salesforce_rtem	api_activity
salesforce_rtem	authentication
salesforce_rtem	datastore_activity
salesforce_rtem	file_hosting_activity
sentinelone_endpointsecurity	dns_activity
sentinelone_endpointsecurity	file_system_activity
sentinelone_endpointsecurity	http_activity
sentinelone_endpointsecurity	process_activity
servicenow_cmdb	api_activity
servicenow_cmdb	datastore_activity
servicenow_cmdb	entity_management
slack_auditlogs	account_change
slack_auditlogs	authentication

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
slack_auditlogs	detection_finding
slack_auditlogs	entity_management
slack_auditlogs	file_hosting_activity
slack_auditlogs	user_access_management
slack_auditlogs	web_resources_activity
snowflake_platform	authentication
snowflake_platform	datastore_activity
snowflake_platform	entity_management
snowflake_platform	group_management
snowflake_platform	user_access_management
snowflake_platform	user_inventory_info
tanium_endpointmanagement	account_change
tanium_endpointmanagement	authentication
tanium_endpointmanagement	compliance_finding
tanium_endpointmanagement	detection_finding
tanium_endpointmanagement	device_inventory_info
tanium_endpointmanagement	entity_management
tanium_endpointmanagement	group_management
tanium_endpointmanagement	vulnerability_finding
tenable_vulnerabilitymanagement	account_change

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
tenable_vulnerabilitymanagement	api_activity
tenable_vulnerabilitymanagement	authentication
tenable_vulnerabilitymanagement	compliance_finding
tenable_vulnerabilitymanagement	device_inventory_info
tenable_vulnerabilitymanagement	scan_activity
tenable_vulnerabilitymanagement	vulnerability_finding
wazuh_platform	api_activity
wazuh_platform	application_lifecycle
wazuh_platform	authentication
wazuh_platform	compliance_finding
wazuh_platform	detection_finding
wazuh_platform	device_inventory_info
wazuh_platform	file_system_activity
wazuh_platform	remediation_activity
wazuh_platform	vulnerability_finding
wiz_cnapp	api_activity
wiz_cnapp	authentication
wiz_cnapp	compliance_finding
wiz_cnapp	detection_finding
wiz_cnapp	vulnerability_finding

数据源名称（@data_source_name 字段）	数据源类型（@data_source_type 字段）
zeek	authentication
zeek	base_event
zeek	detection_finding
zeek	dhcp_activity
zeek	dns_activity
zeek	email_activity
zeek	ftp_activity
zeek	http_activity
zeek	network_activity
zeek	rdp_activity
zeek	smb_activity
zeek	software_inventory_info
zeek	ssh_activity
zeek	tunnel_activity
zscaler_internetaccess	authentication
zscaler_internetaccess	dns_activity
zscaler_internetaccess	http_activity
zscaler_internetaccess	network_activity

其他第三方来源 AWS Security Hub CSPM

其他第三方安全发现可通过 AWS Security Hub CSPM 集成获得。以下合作伙伴将调查结果发送到 Security Hub CSPM，然后这些结果将作为数据源出现在日志中 CloudWatch 。有关这些集成的全面详细信息，请参阅《Security Hub 用户指南》中的 S ecurity Hub CSPM Third-party 产品集AWS 成。

Partner	集成
3CoreSec — NTA	通过 Security Hub CSPM 发送调查结果
警报逻辑 — SiemLess 威胁管理	通过 Security Hub CSPM 发送调查结果
Aqua Security — 云原生安全平台	通过 Security Hub CSPM 发送调查结果
水族安全 — Kube-bench	通过 Security Hub CSPM 发送调查结果
盔甲 — 任何地方的盔甲	通过 Security Hub CSPM 发送调查结果
AttackIQ	通过 Security Hub CSPM 发送调查结果
梭子鱼网络 — 云安全守护者	通过 Security Hub CSPM 发送调查结果
BigID — BigID Enterprise	通过 Security Hub CSPM 发送调查结果
蓝色六边形	通过 Security Hub CSPM 发送调查结果
检查点 — CloudGuard IaaS	通过 Security Hub CSPM 发送调查结果
检查点 — CloudGuard 姿势管理	通过 Security Hub CSPM 发送调查结果

Partner	集成
Claroty — xDome	通过 Security Hub CSPM 发送调查结果
云存储安全 — 适用于 Amazon S3 的防病毒软件	通过 Security Hub CSPM 发送调查结果
对比安全 — 对比度评估	通过 Security Hub CSPM 发送调查结果
CrowdStrike — CrowdStrike 猎鹰	通过 Security Hub CSPM 发送调查结果
CyberArk — 特权威胁分析	通过 Security Hub CSPM 发送调查结果
数据定理	通过 Security Hub CSPM 发送调查结果
德拉塔	通过 Security Hub CSPM 发送调查结果
Forcepoint — CAS	通过 Security Hub CSPM 发送调查结果
Forcepoint — 云安全网关	通过 Security Hub CSPM 发送调查结果
原力点 — DLP	通过 Security Hub CSPM 发送调查结果
Forcepoint — NGFW	通过 Security Hub CSPM 发送调查结果
赋格曲	通过 Security Hub CSPM 发送调查结果
Guardicore — Centra	通过 Security Hub CSPM 发送调查结果

Partner	集成
HackerOne — 漏洞情报	通过 Security Hub CSPM 发送调查结果
jFrog — X 射线	通过 Security Hub CSPM 发送调查结果
瞻博网络 — vSRX 下一代防火墙	通过 Security Hub CSPM 发送调查结果
k9 安全 — 访问分析器	通过 Security Hub CSPM 发送调查结果
蕾丝工艺	通过 Security Hub CSPM 发送调查结果
McAfee — MVISION CNAPP	通过 Security Hub CSPM 发送调查结果
NETSCOUT — 网络调查员	通过 Security Hub CSPM 发送调查结果
Orca — 云安全平台	通过 Security Hub CSPM 发送调查结果
帕洛阿尔托网络 — Prisma 云计算	通过 Security Hub CSPM 发送调查结果
帕洛阿尔托网络 — Prisma 云企业版	通过 Security Hub CSPM 发送调查结果
Plerion — 云安全平台	通过 Security Hub CSPM 发送调查结果
徘徊者	通过 Security Hub CSPM 发送调查结果
Qualys — 漏洞管理	通过 Security Hub CSPM 发送调查结果

Partner	集成
Rapid7 — InsightVM	通过 Security Hub CSPM 发送调查结果
SentinelOne	通过 Security Hub CSPM 发送调查结果
Snyk	通过 Security Hub CSPM 发送调查结果
Sonrai 安全 — Sonrai Dig	通过 Security Hub CSPM 发送调查结果
Sophos — 服务器保护	通过 Security Hub CSPM 发送调查结果
StackRox — Kubernetes 安全	通过 Security Hub CSPM 发送调查结果
Sumo Logic — 机器数据分析	通过 Security Hub CSPM 发送调查结果
赛门铁克-云工作负载保护	通过 Security Hub CSPM 发送调查结果
Tenable.io	通过 Security Hub CSPM 发送调查结果
Trend Micro – Cloud One	通过 Security Hub CSPM 发送调查结果
Vectra — Cognito Detect	通过 Security Hub CSPM 发送调查结果
Wiz	通过 Security Hub CSPM 发送调查结果
Caveonix — Caveonix Cloud	通过 Security Hub CSPM 发送和接收调查结果

Partner	集成
Cloud Custodian	通过 Security Hub CSPM 发送和接收调查结果
DisruptOps	通过 Security Hub CSPM 发送和接收调查结果
Kion	通过 Security Hub CSPM 发送和接收调查结果
Turbot	通过 Security Hub CSPM 发送和接收调查结果

> **Note**
>
> 此列表反映了在撰写本文时发送调查结果的 Security Hub 合作伙伴集成。由于 S AWS ecurity Hub 会定期添加新的合作伙伴集成，因此请参阅 Sec urity Hub 用户指南中与 S AWS ecurity Hub CSPM Third-party 的产品集成，了解最新的可用合作伙伴列表。

自定义来源

对于自定义应用程序日志，您可以使用以下方法之一定义自己的数据源分类：

- 日志组标签-使用密钥向日志组添加标签，`cw:datasource:name`并为`cw:datasource:type`日志组中提取的所有日志指定数据源名称和类型。
- 管道配置-在接收应用程序日志时，通过日志处理管道配置数据源信息。

> **Note**
>
> 为避免与 AWS 服务日志发生冲突，自定义数据源名称不能以 “aws” 或 “amazon” 开头。

使用日志分析分析 CloudWatch 日志数据

日志分析是 Amazon Lo CloudWatch gs 中的统一控制台体验，它将日志分析功能集中在一个地方。您可以使用 Logs Insights 查询和分析 CloudWatch 日志数据，使用 Live Tail 实时流式传输日志，并通过 Contributor Insights 确定您的 CloudWatch 主要贡献者。日志分析是在日志控制台中分析 CloudWatch 日志的默认体验。

你可以用日志分析做什么

日志分析汇集了以下功能：

- 查询和分析日志数据-使用 Logs Insights 对您的 CloudWatch 日志组运行查询。你可以在不同的选项卡中运行多个查询，保存和重复使用带参数的查询，使用分面浏览数据，使用自然语言生成查询，以及可视化结果。有关更多信息，请参阅 使用“日志见解”分析 CloudWatch 日志数据。
- 近乎实时地流式传输日志 — 使用 Live Tail 查看采集日志事件时的流式传输列表，并筛选或突出显示对您重要的事件。有关更多信息，请参阅 使用 L CloudWatch ogs Live Tai。
- 确定最重要的贡献者 — 使用“ CloudWatch贡献者见解”来分析高基数日志数据，并查看对您的系统影响最大的贡献者。有关更多信息，请参阅使用贡献者见解来分析高基数数据。

开始使用日志分析

要使用日志分析，请登录CloudWatch 控制台，然后在导航窗格的“日志”下选择“日志分析”。

日志分析是默认体验。如果您选择退出，则可以继续使用 L CloudWatch ogs Insights、Live Tail 和 Contributor Insights 作为与众不同的体验以及日志分析，并且可以随时切换回日志分析。

日志分析使用的定价与其底层功能相同： CloudWatch 日志见解查询、实时跟踪和贡献者见解。有关定价的详细信息，请参阅 CloudWatch 定价。

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。除了使用日志组进行查询外，您还可以使用分面、数据源和数据类型进行查询。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。每个账户仅限 100 个并发 Lo CloudWatch gs Insights QL，包括添加到仪表板的查询。此外，您可以为 OpenSearch 服务 PPL 或 OpenSearch 服务 SQL 运行 15 个并发查询。

CloudWatch Logs Insights 支持 3 种可用于查询的查询语言：

- 一种专门构建的 Logs Insights 查询语言（Logs Insights QL），包含一些简单但功能强大的命令。
- OpenSearch 服务管道处理语言 (PPL)。 OpenSearch PPL 允许您使用一组由竖线 (|) 分隔的命令来分析日志。

  借 OpenSearch 助 PPL，您可以使用串接在一起的命令来检索、查询和分析数据，从而更容易理解和撰写复杂的查询。该语法使命令链能够转换和处理数据。使用 PPL，您可以对数据进行筛选和聚合，并使用一组丰富的数学、字符串、日期、条件和其他函数进行分析。
- OpenSearch 服务结构化查询语言 (SQL)。使用 OpenSearch SQL 查询，您可以以声明方式分析日志。您可以使用 SELECT、FROM、WHERE、GROUP BY、HAVING 等命令，以及 SQL 中提供的各种其他命令和函数。您可以跨日志组执行 JOIN 操作，使用子查询关联日志中的数据，并使用丰富的 JSON、数学、字符串、条件和其他 SQL 函数集对日志执行强大的分析。

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

CloudWatch Logs Insights 提供以下功能，可用于任何查询语言。

- 自动发现 Amazon Route 53、 AWS Lambda AWS CloudTrail、和 Amazon VPC 等 AWS 服务的日志中的日志字段，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。
- 创建字段索引可以降低成本并加快查询速度，尤其适用于处理大量日志组或日志事件的查询。为日志事件中常见的字段创建字段索引后，可以在查询中使用它们。查询会跳过处理已知不包含索引字段的日志事件，并处理较少的数据。

> **Note**
>
> `filterIndex` 命令仅在 Logs Insights QL 中可用。

- 检测和分析日志事件中的模式。模式是在日志字段中重复出现的共有的文本结构。查看查询结果时，可以选择 “模式” 选项卡，查看 CloudWatch Logs 根据结果样本找到的模式。
- 保存查询、查看查询历史记录、重新运行已保存的查询，以及使用带参数的已保存查询。
- 向控制面板添加查询。
- 使用加密查询结果。 AWS Key Management Service
- 使用自然语言@@ 生成查询允许您使用自然语言创建 Logs Ins CloudWatch ights 查询。您可以询问有关您要查找的数据的问题或描述该数据，然后 AI 会根据您的提示生成查询，并逐行说明查询的工作原理。
- 使用分面对日志进行分组、筛选和交互式浏览。
- 周围的日志允许您查看查询结果中任何特定日志记录之前和之后的日志行，以获取有关关键日志事件的即时上下文。您可以将范围配置为查看选定记录前后的 5、10、20、50 或 100 条日志行，并在周围的日志中搜索特定的关键字。

只有当您使用 L CloudWatch ogs Insights QL 时，才支持以下日志见解功能。

- 比较查询，将日志组中的日志事件与先前时间段的日志事件进行比较。

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

如果您登录的账户设置为 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 跨账户可观测性。

使用 Logs Insights QL 创建查询时，也可以使用自然语言创建 Logs Ins CloudWatch ights 查询。要执行此操作，请询问或描述您要查找的数据。此 AI-assisted 功能可根据您的提示生成查询，并逐行说明查询的工作原理。有关更多信息，请参阅使用自然语言生成和更新 L CloudWatch ogs Insights 查询。

如果使用任何支持的查询语言的查询尚未完成，则会在 60 分钟后超时。查询结果在 7 天内可用。

CloudWatch 无论查询语言如何，Logs Insights 查询都会根据扫描的未压缩日志数据量收取费用。有关更多信息，请参阅 Amazon CloudWatch 定价。

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

内容

- 支持的查询语言
- 使用自然语言生成和更新 L CloudWatch ogs Insights 查询
- 支持的日志和发现的字段
- 创建字段索引以提高查询性能并减少扫描量
- 使用分面对日志进行分组和浏览
- 在“日志见解”中查看周围的 CloudWatch 日志
- 模式分析
- 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
- 根据 Lo CloudWatch gs Insights 查询结果生成自然语言摘要

支持的查询语言

以下部分列出了每种查询语言支持的命令。它们还介绍语法格式并提供示例查询。

主题

- CloudWatch 日志见解查询语言（Logs Insights QL）
- OpenSearch 管道处理语言 (PPL)

- OpenSearch 结构化查询语言 (SQL)

CloudWatch 日志见解查询语言（Logs Insights QL）

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

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

主题

- CloudWatch 日志见解语言查询语法
- 开始使用 Logs Insights QL：查询教程
- 查询示例
- 与之前的时间范围进行比较（差异）
- 在图形中可视化日志数据

CloudWatch 日志见解语言查询语法

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

> **Important**
>
> 为了避免因运行大型查询而产生过高的费用，请牢记以下最佳实践：
>
> - 仅为每个查询选择必要的日志组。
> - 始终为查询指定尽可能窄的时间范围。
> - 使用控制台运行查询时，请在关闭 L CloudWatch ogs Insights 控制台页面之前取消所有查询。否则，查询将继续运行直至完成。
> - 向仪表板添加 CloudWatch Logs Insights 小组件时，请确保仪表板刷新频率不高，因为每次刷新都会启动一个新的查询。

要创建包含多个命令的查询，请使用竖线字符（|）分隔命令。

要创建包含注释的查询，请使用哈希字符（#）对注释进行分隔。

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

addtotals	计算数值字段的行合计和列合计。
anomaly	使用机器学习识别日志数据中的异常模式。
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。
filterIndex	强制查询尝试仅扫描在字段索引中提及的字段上编入索引且包含该字段索引值的日志组。这样可以减少扫描量，因为它只尝试扫描这些日志组中包含此字段索引的查询中指定的值的日志事件。 不频繁访问日志类中的日志组不支持此命令。
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	将在您请求的时间段内找到的日志事件与之前相同长度的时间段内的日志事件进行比较，以便您可以查找趋势并确定某些日志事件是否是新的。

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。
SOURCE	SOURCE在查询中包含是根据日志组名称前缀、账户标识符、日志组类别、数据源或日志组标签指定要包含在查询中的大量日志组的有用方法。只有在 AWS CLI 或中以编程方式创建查询时，才支持此命令，而不是在 CloudWatch 控制台中创建查询。
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。limit any用于在找到足够结果后提早停止扫描。
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。
unnest	对输入列表进行扁平化以生成多条记录，列表中的每个元素都有一条记录。
lookup	通过匹配字段值，使用查找表中的数据丰富日志事件。使用查找表将用户详细信息、应用程序名称或产品信息等参考数据添加到查询结果中。
join	根据匹配的字段，将来自源日志组的日志事件与来自另一个日志组的事件或查询结果相结合。使用 join 命令使用不同来源的通用密钥（例如匹配的请求标识符或事务 ID）关联不同来源的相关日志事件。
subqueries	子查询是一种嵌套的 Logs Insights 查询，可用作其他查询的输入。子查询可用于派生中间结果集，然后由后续命令使用。
autoregress	创建字段值的延迟（上一行）副本。
logcompare	将当前时间窗口与向后移动持续时间的基线窗口进行比较。
filldown	向前移动最后一个非空值以填补空白。
fillmissing	在 bin () 的统计数据之后插入空时间箱的行，可以选择用常量填充字段。

cidrlookup	通过将 IP 字段与查找表中的 CIDR 范围进行匹配来丰富事件。
outlier	根据四分位数范围 (IQR) 检测统计异常值，并可以移除或转换异常值行。
accum	计算数值字段的连续累积和。
appendcols	通过位置行匹配将子查询中的列追加到当前结果中。
sessionize	按身份字段和闲置间隙将事件分组到会话中。
countFrequent	返回每个唯一字段值组合的近似计数，按降序排序。
where	充当`filter`命令的语法别名并接受相同的语法。
估计	返回查询在不运行查询的情况下将在选定的日志组和时间范围内扫描的估计字节。
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

- 日志类中支持的 Logs Insights QL 命令
- addtotals
- 异常
- display
- fields
- 筛选
- filterIndex
- SOURCE
- 模式
- diff
- parse
- 相关字段
- 扩展

- 排序
- stats
- limit
- dedup
- unmask
- unnest
- lookup
- join
- subqueries
- 自动倒退
- 日志比较
- 填充
- 填写缺失
- cidrlookup
- 异常值
- accum
- appendcols
- 会话化
- 频繁计数
- where
- 估算
- 布尔值、比较、数值、日期时间和其他函数
- 哈希函数
- Time-series 函数
- 包含特殊字符的字段
- 在查询中使用别名和注释

日志类中支持的 Logs Insights QL 命令

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

addtotals

addtotals用于计算数值字段的行合计和列合计。行总数是按区块计算的。列总数 (col=true) 是在合并后计算的，以确保分布式区块的正确性。您可以使用自定义字段选择、自定义字段名和row=false来禁用行总计。

语法

```
| addtotals [fieldname=Name] [row=bool] [col=bool] [field1, field2, ...]
```

- fieldname=*Name*— 设置行合计列的名称。默认值为 Total。
- row=*bool*— 启用或禁用行合计。默认值为 true。
- col=*bool*— 启用或禁用列合计，这会在底部添加一个摘要行。默认值为 false。
- *field1*, *field2*, ...— 要汇总的特定字段的可选列表。如果省略，则对所有数值字段进行求和。

示例

```
# Row totals (default) - adds a Total column summing all numeric fields
fields a, b
| addtotals
```

```
# Sum only specific fields
fields price, tax, qty
| addtotals price, tax
```

```
# Custom total column name
fields val
| addtotals fieldname=RowSum
```

```
# Column totals - adds a Summary row at the bottom
fields x
| addtotals col=true x
```

异常

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

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

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

anomaly 命令输入

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

已识别的异常类型

anomaly 命令可识别五种不同类型的异常：

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

anomaly 命令输入

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

示例

以下命令可识别日志数据中的模式，然后检测这些模式中的异常：

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

anomaly 命令可以与筛选结合使用，以专注于特定的日志类型：

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

```
| anomaly
```

anomaly 命令可以与排序结合使用以整理结果：

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

display

使用 display 在查询结果中显示一个或多个特定字段。

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

示例：显示一个字段

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

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

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

fields

使用 fields 在查询结果中显示特定字段。

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

示例：显示特定字段

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

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

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

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

筛选

使用 filter 获取与一个或多个条件匹配的日志事件。

示例：使用一个条件筛选日志事件

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

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

示例：使用多个条件筛选日志事件

您可以使用关键字 and 和 or 组合多个条件。

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

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

```
| limit 20
```

索引字段和 filter 命令

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

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

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

示例：使用索引查找与特定请求 ID 相关的日志事件

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

```
fields @timestamp, @message
| filter requestId = "1234656"
| limit 20
```

筛选命令中的匹配项和正则表达式

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

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

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

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

示例：匹配子字符串

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

第一个示例使用 like 匹配子字符串。

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

第二个示例使用 like 和正则表达式模式匹配子字符串。

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

第三个示例使用正则表达式匹配子字符串。

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

示例：使用通配符匹配子字符串

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

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

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

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

可能的匹配项可以采用以下格式：

- ServiceLogSampleApiLogGroup
- SampleApiLogGroupServiceLog

示例：从匹配项中排除子字符串

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

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

示例：使用不区分大小写的模式匹配子字符串

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

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

filterIndex

使用 filterIndex 可以强制查询仅扫描在查询中指定的字段上编制索引的日志组，从而返回已编制索引的数据 对于已在此字段上编制索引的日志组，它会进一步优化查询，跳过没有任何日志事件包含索引字段的查询中指定的字段的日志组。这样可以进一步减少扫描量，因为它会尝试仅扫描这些日志组中与此字段索引的查询中指定的值匹配的日志事件。有关字段索引和如何创建它们的更多信息，请参阅创建字段索引以提高查询性能并减少扫描量。

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

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

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

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

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

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

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

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

CloudWatch 日志还为某些数据源名称和类型组合提供默认字段索引。默认字段索引自动适用于以下数据源名称和类型组合：

数据源名称和类型	默认字段索引
amazon_vpc.flow	action logStatus region flowDirection

数据源名称和类型	默认字段索引
	type
amazon_route53.resolver_query	query_type transport rcode
aws_waf.access	action httpRequest.country
aws_cloudtrail.data aws_cloudtrail.management	eventSource eventName awsRegion userAgent errorCode eventType managementEvent readOnly eventCategory requestId

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

filterIndex 与 filter 之比较

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

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

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

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

SOURCE

SOURCE在使用 AWS CLI 或 API 创建查询时，在查询中包含是指定要包含在查询中的日志组 and/or 数据源的有用方法。只有 AWS CLI 和 API 支持该SOURCE命令， CloudWatch 控制台不支持该命令。使用 CloudWatch控制台启动查询时，您可以使用控制台界面来指定日志组。

查询日志组

要使用 SOURCE 指定要查询的日志组，可以使用以下关键字：

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

  列表中最多可以包含 5 个前缀。

- `accountIdentifier`对指定 AWS 账户中的日志组运行查询。仅当您在监控账户中运行查询时，这才有效。如果省略此项，则默认为查询所有关联的源账户和当前的监控账户。有关跨账户可观察性的更多信息，请参阅CloudWatch 跨账户可观察性。

  列表中最多可以包含 20 个账户标识符。

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

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

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

```
SOURCE logGroups()
```

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

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

下一个示例根据名称前缀选择日志组。

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

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

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

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

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

最后一个示例显示了如何在SOURCE命令中使用该`start-query` AWS CLI 命令。

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

查询数据源

SOURCE要用于指定要查询的数据源，可以使用dataSource关键字。列表中最多可以包含十个数据源。

以下示例选择amazon_vpc.flow数据源。

```
SOURCE dataSource(['amazon_vpc.flow'])
```

以下示例选择amazon_vpc.flow数据源并根据日志组名称前缀限制日志组。

```
SOURCE dataSource(['amazon_vpc.flow']) logGroups(namePrefix: ['namePrefix1'])
```

按标签查询日志组

SOURCE要使用按标签筛选日志组，请使用logGroupTags函数。将标签指定为标签过滤器列表，每个过滤器都有一个key可选values数组。

- 使用不同键的多个标签过滤器与 AND 逻辑相结合。
- 同一标签筛选器中的多个值与 OR 逻辑相结合。
- *用于通配符匹配。例如，payment*匹配以开头的值payment。
- !用作否定的前缀。例如，!production匹配不匹配的值production。
- 您可以包含多达 5 个标签过滤器，每个过滤器最多包含 5 个值。

以下示例选择所有标记为的日志组team=team1 OR team=team2。

```
SOURCE logGroupTags([{"key":"team", "values":["team1", "team2"]}])
| fields @message, @timestamp
```

下一个示例选择service标签开头的日志组payment，而environment标签不是production。

```
SOURCE logGroupTags([{"key":"service", "values":["payment*"]}, {"key":"environment",
 "values":["!production"]}])
| fields @message, @timestamp
```

以下示例将标签筛选与名称前缀过滤器结合在一起。

```
SOURCE logGroups(namePrefix: ['/aws/lambda']) logGroupTags([{"key":"environment",
 "values":["production"]}])
| fields @message, @timestamp
```

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

模式命令输入

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

如 CloudWatch 果 Logs 无法推断出动态令牌所代表的数据类型，则将其显示为 <Token-*number* >，并*number*指明与其他动态令牌相比，该标记在模式中的显示位置。

动态令牌的常见示例包括错误代码、IP 地址、时间戳以及请求 ID。

模式命令输出

pattern 命令将生成以下输出：

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

  CloudWatch Logs 根据对包含该名称的日志事件内容的分析来分配名称的字符串部分。

  如 CloudWatch 果 Logs 无法推断出动态令牌所代表的数据类型，则将其显示为 <Token-*number* >，并*number*指明与其他动态令牌相比，该标记在模式中的显示位置。

  例如，[INFO] Request time: <Time-1> ms 是日志消息 [INFO] Request time: 327 ms 的潜在输出。
- @ratio：选定时间段的日志事件与符合已确定模式的指定日志组的比率。例如，如果所选日志组和时间段中有一半的日志事件符合模式，则 @ratio 返回 0.50
- @sampleCount：选定时间段的日志事件与符合已确定模式的指定日志组的数量。
- @severityLabel：日志严重性或级别，表示日志中包含的信息类型。例如，Error、Warning、Info 或 Debug。

示例

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

```
pattern @message
```

pattern 命令可以与 filter 命令结合使用

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

pattern 命令可以与 parse 和 sort 命令结合使用

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

diff

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

在 diff 命令中添加修饰符以指定要比较的时间段：

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

有关更多信息，请参阅 与之前的时间范围进行比较（差异）。

parse

parse用于从日志字段中提取数据并创建可在查询中处理的提取字段。该parse命令支持四种模式：glob 表达式、正则表达式、logfmt 和 CSV。

fieldName如果省略，@message则默认使用。通过将字段名指定为第一个参数，可以从任何命名字段中进行解析。

如果日志事件与指定的模式不匹配，您仍会在结果中看到该事件，但不会显示提取的字段。

Glob 模式

使用通配符 (*) 作为要提取的值的占位符，然后使用将其分配给命名字段。as

语法

```
parse fieldName "pattern" as alias1, alias2
```

*通配符的数量必须等于别名的数量。

示例

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

```
parse @logStream "*/*/*/*" as env, service, instance, shard
| stats count(*) by env, service
```

链式解析

提取一个字段，然后进一步解析提取的字段。

```
parse @message "url=*" as url
| parse url "/api/*/users/*" as apiVersion, userId
| display apiVersion, userId
```

正则表达式模式

使用带有命名捕获组的正则表达式来提取字段。有关正则表达式语法的信息，请参阅 支持的正则表达式（regex）语法。

语法

```
parse fieldName /regex/
```

使用命名的捕获组(?<name>...)来定义提取的字段。

示例

使用命名的捕获组提取字段

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

使用命名的捕获组从 VPC 流日志中提取 ENI

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

Multi-match mode

使用多重匹配模式从字段中提取正则表达式的所有匹配项，从而为每个日志事件生成多行。在正则表达式`multi`模式后面添加关键字。

语法

```
parse fieldName /regex/ multi
```

示例

从日志行中提取所有 IP 地址（多重匹配）

```
parse @message /(\d+\.\d+\.\d+\.\d+)/ as ip_addr multi
| stats count(*) by ip_addr
```

logfmt 模式

用于`parse logfmt`将 logfmt 格式的日志行解析为键值对。Logfmt 是一种结构化日志格式，其中每行都包含以空格`key=value`分隔的对。

语法

```
parse fieldName logfmt as alias
```

结果是您使用点表示法访问的地图（例如`lf.level`，`lf.msg`）。

示例

```
parse @message logfmt as lf
| filter lf.level = "error"
| display lf.msg, lf.duration
```

```
parse @message logfmt as lf
| stats count(*) by lf.host
```

CSV 模式

用于`parse csv`将 CSV-formatted 日志行解析为结构化字段。每个以逗号分隔的值都分配给相应的别名。

语法

```
parse fieldName csv as alias1, alias2, alias3
```

示例

```
parse @message csv as timestamp, level, message
| filter level = "ERROR"
| display timestamp, message
```

```
parse @message csv as host, method, path, status, duration
| stats avg(duration) by method
```

JSON 字段提取

`json field=fieldName`用于从先前解析的对象字段中进行显式链式 JSON 提取。这使您无需重新解析原始消息即可从结构化字段中提取嵌套密钥。

语法

```
json field=fieldName "key.subkey" as alias
```

示例

```
parse @message /(?<payload>\{.*\})/ as payload
```

```
| json field=payload "user.name" as username
| display username
```

```
json field=requestContext "identity.sourceIp" as caller_ip
| stats count(*) by caller_ip
```

相关字段

relevantfields用于自动识别日志数据中哪些字段与给定条件最相关。该命令将基准日志与符合条件的子集进行比较，并返回按相关性分数排名的字段。

语法

```
relevantfields [field1, field2, ...] where condition
```

字段列表是可选的。如果省略，则会分析所有字段。

该命令返回以下输出字段：

- @fieldName— 字段的名称
- @relevanceScore— 以 0 到 1 的等级表示相关性的分数
- @topRelevanceContributors— 对于分类字段，显示频率偏移最高的条目。
- @conditionalMedian— 对于数值字段，日志中符合条件的值的中位数
- @baselineMedian— 对于数值字段，日志中不符合条件的值的中位数

示例

分析所有字段的 API 响应速度是否很慢

```
relevantfields where Time > 400
```

重点分析特定领域

```
relevantfields Controller, Path, Service where Time > 400
```

识别 Lambda 超时的驱动因素

```
relevantfields where Duration > 5000
```

扩展

expand用于获取一个包含 JSON 数组的字段，并为数组中的每个元素创建一个单独的日志事件。原始日志事件中的所有其他字段都会在每个新事件中重复。

语法

```
expand fieldName
```

示例

如果日志事件包含items = ["apple","banana","cherry"]和host = "web-01"，则expand items会生成三个日志事件：{items: "apple", host: "web-01"}{items: "banana", host: "web-01"}、和{items: "cherry", host: "web-01"}。

```
expand items
| stats count(*) by items, host
```

排序

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

该排序算法是自然排序的更新版本。如果按升序排序，则使用以下逻辑。

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

有关 Unicode 顺序的更多信息，请参阅 List of Unicode characters。

例如，以下是按升序排序的结果。

```
!: >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> sorted by unicode order
#
*%04
```

```
0# >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Alphanumeric starting with numbers
5A
111A   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  Starts with more digits than 5A, so it sorted
 to be later than 5A
2345_
@ >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 2345 is compared with @ in the unicode order,
@_
A >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Values starting with letters
A9876fghj
a12345hfh
0 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Number values
01
1
2
3
```

如果按降序进行排序，则排序结果是相反的。

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

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

stats

使用 `stats` 创建日志数据的可视化效果，例如条形图、折线图和堆叠面积图。这可以帮助您更有效地识别日志数据中的模式。 CloudWatch Logs Insights 为使用该`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)
```

以下查询收集每项服务的不同状态值。

```
stats values(status) as statuses by service
```

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个stats命令。这使您能够对第一次聚合的输出执行其他聚合。查询中允许的最大stats命令数取决于日志组的日志类别。

示例：使用两个 **stats** 命令进行查询

例如，以下查询首先找到 5 分钟分箱中的总流量，然后计算这 5 分钟分箱中的最高、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fields中将多个stats命令与其他命令（例如filter和）组合在一起。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 countDistinct(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 10 个stats命令。无法更改此配额。

如果您使用sort或limit命令，则它必须出现在最后一个stats命令之后。如果它在最后一个stats命令之前，则查询无效。

当一个查询有多个stats命令时，查询的部分结果要等到第一次stats聚合完成后才会开始显示。

在单个查询的后续stats命令中，您只能引用前面的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stats命令之后引用的任何字段都必须在该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，如果不使用前面的stats命令传播timestamp字段，就无法bin在后续stats命令中使用。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
```

取而代之的是，在前面的stats命令中定义该@timestamp字段，然后可以在后续stats命令dateceil中使用该字段，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	说明
avg(fieldName: NumericLogField)	数字	指定的字段中值的平均值。
count() count(fieldName: LogField)	数字	计算日志事件的数量。count()（或 count(*)）对查询返回的所有事件进行计数，而 count(fieldName) 对包含所指定字段名称的所有记录进行计数。
countDistinct(fieldName: LogField)	数字	返回字段的唯一值的数量。如果字段具有非常高的基数（包含许多唯一值），则 countDistinct 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。
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)	数字	指定的字段中值的标准偏差。
sum(fieldName: NumericLogField)	数字	指定的字段中值的总和。
values(fieldName: LogField)	array	为每个组收集指定字段的不同值。也称为 COLLECT_VALUES 。
variance(fieldName: NumericLogField)	数字	指定字段中值的总体方差。

函数	结果类型	说明
		示例：`stats variance(responseTime) as rt_var by service`
`topk(k: number, fieldName: LogField)`	array	返回指定字段中最常用的 k 个值。k 的有效值范围为 1 到 10000。此函数不能与其他聚合函数组合使用by或其他聚合函数。 示例：`stats topk(3, status)`

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	说明
`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。

limit

使用 `limit` 指定您希望查询返回的日志事件数。如果省略`limit`，则查询默认为最多返回 10,000 个日志事件。您最多可以指定为 100,000 的`limit`值。

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
```

限制任何

`limit any N`用于在找到*N*匹配的日志事件后立即停止查询，而不扫描剩余数据。当您只需要匹配事件的样本而不是特定的有序集时，这可以显著减少扫描的数据量并加快查询速度。

使用常规查询时`limit`，查询会扫描时间范围内的所有数据，然后返回排名靠前的*N*结果。使用`limit any`，一旦找到足够的结果，查询就会提前完成，这意味着返回的事件可能来自时间范围的任何部分。

例如，以下查询返回 1 个日志事件，并在找到匹配项后立即停止扫描：

```
fields @message | limit any 1
```

> **Note**
>
> 由于扫描`limit any`会提早停止，因此不保证结果会按任何特定的顺序排列。如果您需要有序的结果，请`any`将`limit`不与一起使用`sort`。

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`。

当您在查询dedup中使用时，控制台会显示一条消息，例如显示 Y 条记录中的 X，其中 X 是已删除重复数据的结果数，Y 是重复数据删除前匹配的记录总数。这表示重复的记录已被删除，但并不意味着数据丢失。

示例：仅查看名为 **`server`** 的字段的每个唯一值的最新日志事件

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

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

有关 CloudWatch Logs Insights 查询的更多示例，请参阅常规查询。

unmask

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

有关日志组中数据保护的更多信息，请参阅 通过屏蔽帮助保护敏感的日志数据。

unnest

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

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

命令结构

以下示例描述了此命令的格式。

```
unnest field into unnested_field
```

示例查询

以下示例解析了 JSON 对象字符串并展开字段事件列表。

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

此示例查询的日志事件可能是一个 JSON 字符串，如下所示：

```
{
   "events": [
        {
            "name": "exception"
```

```
        },
        {
            "name": "user action"
        }
    ]
}
```

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

示例查询

以下示例会对列表进行扁平化，然后筛选出项目。

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

示例查询

以下示例会对列表进行扁平化以进行聚合。

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

lookup

使用lookup查找表中的参考数据来丰富查询结果。您可以使用上传到 Amazon Logs 的 CSV 数据或已完成或已取消的 CloudWatch 日志查询的结果填充查询表。运行查询时，该lookup命令会将日志事件中的字段与查找表中的字段进行匹配，并将指定的输出字段附加到结果中。

对数据丰富场景使用查找表，例如将用户 ID 映射到用户详细信息，将产品代码映射到产品信息，或者将错误代码映射到错误描述。

创建和管理对照表

必须先创建对照表，然后才能在查询中使用该lookup命令。您可以通过 CloudWatch 控制台或使用 Amazon L CloudWatch ogs API 创建和管理查询表。

创建对照表（控制台）

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 “设置”，然后选择 “日志” 选项卡。
3. 滚动到 “查找表”，然后选择 “管理”。
4. 选择 “创建对照表”。
5. 输入查找表的名称。名称只能包含字母数字字符和下划线。
6. （可选）输入描述。
7. 上传 CSV 文件。该文件必须包含带有列名的标题行，使用 UTF-8 编码，且不超过 10 MB。
8. （可选）指定用于加密表数据的密 AWS KMS 钥。
9. 选择创建。

创建对照表后，您可以在 L CloudWatch ogs Insights 查询编辑器中查看该表。选择 “查找表” 选项卡以浏览可用表及其字段。

要更新查找表，请选择该表，然后选择操作、更新。上传新的 CSV 文件以替换所有现有内容。要删除对照表，请选择操作，删除。

除了上传 CSV 数据外，您还可以根据您在日志分析控制台或使用 Amazon CloudWatch Logs API 运行的已完成或取消的日志查询的结果创建或更新对照表。使用`CreateLookupTable`或`UpdateLookupTable` API 的`queryId`参数指定查询 ID。取消的查询会使用查询停止时可用的部分结果填充表格。必须指定 CSV 内容或查询 ID，但不能同时指定两者。更新表格是一种完全替换的操作——无论来源如何，所有现有内容都会被替换。要按计划自动刷新包含查询结果的查找表，请使用带有查找表目标的计划查询。有关更多信息，请参阅 为计划查询配置查找表目的地。

> **Note**
>
> 每个账户最多可以创建 100 个查询表 AWS 区域。CSV 文件的最大大小为 10 MB。您也可以使用 Amazon CloudWatch 日志 API 管理查询表。有关更多信息，请参阅 Amazon CloudWatch 日志 API 参考CreateLookupTable中的。

> **Note**
>
> 如果使用 KMS 密钥对查找表进行加密，则调用者必须拥有该密钥（用于加密查找表的 KMS 密钥）的`kms:Decrypt`权限，才能将 `StartQuery` API 用于引用该查找表的查询。有关更多信息，请参阅 使用加密 CloudWatch 日志中的查找表 AWS Key Management Service。

查找的查询语法

命令结构

下面显示了此命令的格式。

```
lookup table match-fields output-mode output-field[,...]
```

该命令使用以下参数：

- *table*-要使用的查找表的名称。
- *match-fields*— 指定一个或多个字段以将日志事件与查找表进行匹配。您可以使用以下任一表格：
  - *lookup-field* as *log-field* [,...]— as 当查找表的列名与日志事件字段名称不同时使用。例如，将查找表中的ip_address列与日志事件中的srcAddr字段进行ip_address as srcAddr匹配。
  - *lookup-field* [,...]— 当日志事件字段名称与查找表列名相同时，您可以省略as并直接指定字段名称。例如，将两列与同名的日志事件字段进行department, role匹配。

  当指定了多个匹配字段时，查找表中的某一行必须匹配所有字段才能生成结果（AND 逻辑）。
- *output-mode*— 指定如何将输出字段添加到结果中。使用以下值之一：
  - OUTPUT— 将输出字段添加到结果中。如果日志事件中已存在同名字段，则该字段将被查找表值覆盖。如果未找到匹配项，则该字段将设置为 null。
  - OUTPUTNEW— 仅当日志事件中尚不存在输出字段时，才将输出字段添加到结果中。如果该字段已有值，则保留原始值。如果未找到匹配项，则该字段保持不变。
- *output-field*— 查找表中要添加到结果中的一个或多个字段。

示例：使用用户详细信息丰富日志事件

假设您有一个日志组，其中包含包含一个id字段的事件，以及一个user_data以idnameemail、和列命名的查找表department。以下查询使用查询表中的用户名、电子邮件和部门来丰富每个日志事件。

```
fields action, status, name, email, department
| lookup user_data id OUTPUT name, email, department
```

示例：使用带聚合的查找

您可以将查找输出字段与聚合函数一起使用。以下查询使用用户详细信息丰富日志事件，然后统计按电子邮件地址分组的事件。

```
fields user_id, action, username, email, department
| lookup user_data user_id OUTPUT username, email, department
| stats count(*) by email
```

示例：使用带过滤器的查找

您可以根据查询返回的字段筛选结果。以下查询丰富了日志事件，然后筛选出仅显示来自特定部门的事件。

```
fields user_id, action
| lookup user_data user_id OUTPUT username, email, department
| filter department = "Engineering"
```

示例：使用 OUTPUTNEW 在不覆盖的情况下进行丰富

如果您的日志事件已经包含一个hostname字段，但该字段有时为空，OUTPUTNEW请使用填写缺失值而不覆盖现有值。

```
fields srcAddr, hostname
| lookup known_hosts ip_address as srcAddr OUTPUTNEW hostname, region
```

示例：使用包含多个匹配字段的查找

您可以在多个字段上进行匹配。以下查询将查询表srcAddr与dstPort对照表进行匹配，以识别已知的网络服务。

```
fields @timestamp, srcAddr, dstAddr, dstPort
| lookup network_services ip_address as srcAddr, port as dstPort OUTPUT service_name,
 owner
| filter ispresent(service_name)
```

示例：使用带有匹配字段名称的查找

当您的日志事件字段名称与查找表的列名完全匹配时，可以省略该as关键字。以下查询直接将department和role字段与对照表进行匹配。

```
fields @timestamp, department, role
| lookup employees department, role OUTPUT office, manager
```

join

根据匹配的字段，将来自源日志组的日志事件与来自另一个日志组的事件或查询结果相结合。

使用join命令将不同来源（例如日志组）的相关日志事件关联起来，使用日志组中通用的密钥（例如匹配的请求标识符或事务 ID）。

语法

```
join type=<join_type> left=<left_alias> right=<right_alias>
    where <left_alias>.<field>=<right_alias>.<field>
    (SOURCE <right_log_group>)
```

参数

- <right_log_group>— 要连接的辅助数据源。
- <left_alias>和 <right_alias>-用于将字段与左（主）和右（辅助）数据源区分开来的别名。
- where <field>— 指定用作联接键的字段。该字段必须存在于两个数据源中。
- type=<join_type>（可选）-指定联接类型。有效值为：
  - inner（默认）-仅返回匹配的记录
  - left— 返回主数据源中的所有记录和辅助数据源的匹配记录

示例

Example示例 1：将 API Gateway 请求与 Lambda 执行日志相关联

此示例说明如何将 API Gateway 访问日志与 Lambda 函数日志结合起来，以将传入的请求与其后端处理关联起来。这对于排除端到端请求流故障以及确定哪些 Lambda 调用对应于特定 API 请求非常有用。

```
filter status >= 500
| join type=inner left=api right=lambda
    where api.requestId=lambda.requestId
    (SOURCE '/aws/lambda/my-function')
| fields api.requestId, api.status, api.latency, lambda.duration, lambda.memoryUsed
| sort api.latency desc
```

这个查询：

1. 查询 API Gateway 访问日志并筛选服务器错误（状态 >= 500）
2. 使用两个日志源中都显示的`requestId`字段与 Lambda 函数日志联接
3. 使用别名（`api`和`lambda`）将字段与每个来源区分开来
4. 返回显示 API 延迟以及 Lambda 执行持续时间和内存使用情况的组合信息
5. 按 API 延迟对结果进行排序以确定最慢的请求

Example示例 2：跨微服务跟踪分布式事务

在微服务架构中调试问题时，通常需要跨多个服务跟踪事务。此示例说明如何使用公共事务 ID 加入来自两个不同服务的日志。

```
filter eventType = "ORDER_CREATED"
| join type=left left=order right=payment
    where order.transactionId=payment.transactionId
    (SOURCE '/aws/lambda/payment-service')
| filter payment.eventType = "PAYMENT_PROCESSED" or !ispresent(payment.eventType)
| fields order.transactionId, order.orderId, order.customerId,
    payment.paymentStatus, payment.amount
| filter payment.paymentStatus != "SUCCESS" or !ispresent(payment.paymentStatus)
```

这个查询：

1. 从订单服务中的订单创建事件开始
2. 使用 a `left join` 来包含所有订单，即使是那些没有匹配付款记录的订单
3. 使用共享`transactionId`字段加入付款处理事件
4. 筛选最终结果以仅显示付款失败或付款记录缺失的订单

左联在此很重要，因为它可以确保您看到已创建但从未发生过相应付款事件的订单，这可能表明系统出现故障。

行为

- 首先处理主数据源（左侧）。
- 使用指定的联接键对辅助数据源进行评估和匹配。
- 使用连接字段上的相等比较来执行匹配。
- 对于左联接，主数据源中不匹配的记录将保留，次要字段的值为空。

注释和限制

- 仅支持相等 (=) 条件。
- 每个查询仅支持一个联接命令。
- 联接密钥必须存在于两个数据源中，并且类型必须兼容。
- 使用 join 的查询可能会扫描更多数据并产生更高的费用。
- 为确保查询性能，辅助数据源中唯一键值的数量限制为 50,000。
- 不支持连接右侧的子查询。

相关命令

- 字段
- 筛选条件
- 解析
- stats
- 排序
- limit

subqueries

子查询是一种嵌套的 Logs Insights 查询，可用作其他查询的输入。子查询可用于派生中间结果集，然后由后续命令使用。

语法

筛选器中的子查询

```
filter <field> in (
    <subquery>
)
```

参数

- `<subquery>`— 返回结果集的有效 Logs Insights 查询。子查询必须生成由外部查询引用的字段。

示例

Example示例 1：查找在下游服务中遇到错误的请求

此示例说明如何使用子查询来识别主服务中导致下游服务出错的请求。这对于排除分布式系统中的级联故障非常有用。

```
filter requestId in (
    SOURCE '/aws/lambda/database-service'
    | filter errorType = "DatabaseConnectionTimeout"
    | fields requestId
)
| fields @timestamp, requestId, endpoint, userId, responseTime
| sort @timestamp desc
```

这个查询：

1. 子查询从数据库服务中查找所有遇到连接超时的`requestId`值
2. 外部查询会筛选主服务的日志，仅显示与那些容易出错的请求 ID 相匹配的请求
3. 结果显示了下游失败的请求的完整背景，包括哪些端点和用户受到影响

这种模式可以帮助您了解下游故障对上游的影响。

Example示例 2：识别经常失败的定向调查请求

此示例演示了使用带聚合功能的子查询来查找重复失败的请求，这通常表示系统性问题而不是暂时性错误。

```
filter requestId in (
    SOURCE '/aws/lambda/payment-processor'
    | filter status = "FAILED"
    | stats count(*) as failureCount by requestId
    | filter failureCount > 3
    | fields requestId
)
| fields @timestamp, requestId, customerId, amount, failureReason
| sort @timestamp asc
```

这个查询：

1. 子查询汇总失败的付款尝试并识别失败次数超过 3 次的请求 ID

2. 外部查询会检索那些有问题的请求 ID 的所有日志事件
3. 结果按时间顺序排序，以显示重试的进度

这有助于区分需要更深入调查的暂时性故障（单次出现）和持续性问题（多次故障）。

行为

- 子查询独立于外部查询执行。
- 结果在被外部查询使用之前被具体化。
- 只有在子查询中明确选择的字段才可供外部查询使用。

注释和限制

- 子查询必须返回由外部查询引用的字段。
- 不支持嵌套子查询。
- 子查询可能会增加查询的执行时间和成本。
- 不支持关联子查询。
- 内部查询执行限制在 30 秒以内。

相关命令

- 字段
- 筛选条件
- 解析
- stats
- 排序
- limit

自动倒退

使用`autoregress`命令创建字段值的延迟（上一行）副本。这 lag/lead 等同于 “ CloudWatch 日志见解”。

语法

```
| autoregress field [AS alias] [p=start[-end]]
```

该命令使用以下参数：

- *field*-要为其创建延迟值的字段。
- AS *alias*（可选）-输出字段的别名。
- p=*N*（可选）-单延迟深度（默认为 1）。p=*N*-*M*用于一定范围的延迟。

示例

以下查询创建了四个延迟字段（bytes_p1直到bytes_p4）。

```
fields @timestamp, bytes
| autoregress bytes p=1-4
```

日志比较

使用logcompare命令将当前时间窗口与向后移动指定持续时间的基线窗口进行比较。

语法

```
| logcompare timeshift duration
```

该命令使用以下参数：

- timeshift *duration*— 基线窗口向后移的持续时间（例如，7d）。

示例

以下查询将过去一天的日志与一周前同期的日志进行比较。

```
SOURCE lg start=-1d end=now
| logcompare timeshift 7d
```

填充

使用filldown命令将最后一个非空值向前移动，以填补空白。您可以指定字段名称或使用通配符。如果未指定字段，则所有字段均已填充。

语法

```
| filldown [field1 [field2 ...]]
```

该命令使用以下参数：

- *field*（可选）-一个或多个字段名称或通配符模式。如果省略，则填充所有字段。

示例

以下查询继承了host字段中的非空值和所有匹配cpu*的字段。

```
fields @timestamp, host, cpu_user, cpu_system
| filldown host cpu*
```

填写缺失

之后stats ... by bin()使用fillmissing命令为空时间箱插入行。您可以选择使用常量值填充字段。

语法

```
| fillmissing [with value for field [, value for field ...]]
```

该命令使用以下参数：

- with *value* for *field*（可选）-分配给插入行中指定字段的常量值。

示例

以下查询为该字段填充空的 1 分钟数据桶，其中 0。avg_latency

```
fields @timestamp, latency
| stats avg(latency) as avg_latency by bin(1m)
| fillmissing with 0 for avg_latency
```

cidrlookup

使用该cidrlookup命令通过将 IP 字段与查找表中的 CIDR 范围进行匹配来丰富事件。

语法

```
| cidrlookup table ipField [AS cidrColumn]
    OUTPUT|OUTPUTNEW lookupField [AS eventField] [, ...]
```

该命令使用以下参数：

- *table*— 包含 CIDR 范围的查找表的名称。
- *ipField*— 包含要匹配的 IP 地址的事件字段。
- AS *cidrColumn*（可选）-查找表中包含 CIDR 范围的列。
- OUTPUT|OUTPUTNEW— OUTPUT 用于覆盖现有字段，或OUTPUTNEW仅填空字段。
- *lookupField*— 查找表中要添加到事件中的一个或多个字段。

示例

以下查询使用网络查询表中的区域、所有者和数据中心丰富了事件。

```
fields @timestamp, srcAddr
| cidrlookup net_table srcAddr OUTPUT region, owner, datacenter
```

异常值

使用outlier命令根据四分位数范围 (IQR) 检测数值字段中的统计异常值。您可以移除或限制异常值行。

语法

```
| outlier [action=remove|transform] [param=n] [uselower=true|false] [mark=true|
false] field [field ...]
```

该命令使用以下参数：

- action（可选）-remove（删除异常值行）或transform（夹住边界）。默认值：transform。
- param（可选）-IQR 乘数。默认值：2.5。
- uselower（可选）-是否检测下限异常值。默认值：false。
- mark（可选）-是否添加布尔标记字段。默认值：false。

- *field*— 用于检查异常值的一个或多个数值字段。

示例

以下查询使用带有下限检测和标记的 3.0 IQR 乘数删除 cpu 或内存为异常值的行。

```
fields @timestamp, cpu, memory
| outlier action=remove param=3.0 uselower=true mark=true cpu memory
```

accum

使用accum命令计算数值字段的累积累和。

语法

```
| accum field [AS out]
```

该命令使用以下参数：

- *field*— 要累积的数值字段。
- AS *out*（可选）-输出字段的别名。

示例

以下查询计算总字节数。

```
fields @timestamp, bytes
| accum bytes AS totalBytes
```

appendcols

使用appendcols命令通过位置行匹配将子查询的列追加到当前结果中。

语法

```
| appendcols [override=true|false] [max=n] ( subquery )
```

该命令使用以下参数：

- override（可选）-是否覆盖现有字段。默认值：false。
- max（可选）-要处理的最大行数（1—100000，默认为 10000）。
- *subquery*— 用括号括起来的完整的“ CloudWatch 日志见解”查询。

示例

以下查询附加了来自另一个日志组的每项服务的平均延迟。

```
fields svc
| stats count(*) as cnt by svc
| appendcols override=true ( SOURCE lg | stats avg(latency) as avgLat by svc )
```

会话化

使用该sessionize命令按身份字段和非活动状态间隙将事件分组为会话。该命令会发出会话标识符字段。

语法

```
| sessionize field[, field ...] [maxspan duration] [AS out]
```

该命令使用以下参数：

- *field*— 一个或多个要分组依据的身份字段。
- maxspan *duration*（可选）-开始新会话之前的最大闲置间隔。默认值：30m。
- AS *out*（可选）-输出字段名称。默认值：session_id。

示例

以下查询按用户和设备将事件分组为会话，闲置间隔为 1 小时。

```
fields @timestamp, userId, deviceId
| sessionize userId, deviceId maxspan 1h as my_session
```

频繁计数

使用countFrequent命令（也可用作count_frequent）计算每个唯一字段值组合的近似计数，按降序排序。该命令会发出一个_approxcount字段。

语法

```
| countFrequent field[, field ...]
```

该命令使用以下参数：

- *field*— 一个或多个用于计算频率的字段。

示例

以下查询计算了方法和状态组合的频率。

```
fields method, status
| countFrequent method, status
```

where

使用该where命令作为filter命令的别名。它接受相同的语法和行为。

语法

```
| where condition
```

该命令使用以下参数：

- *condition*— 与filter命令接受的值相同的布尔表达式。

示例

以下查询过滤器用于筛选包含 “错误” 的日志事件。

```
fields @timestamp, @message
| where @message like /error/
```

估算

使用estimate命令返回查询将在选定的日志组和时间范围内扫描的估计字节，而不运行查询。使用此命令返回的估计字节数，您可以在运行查询之前优化日志组选择、时间范围和过滤器。该estimate命令不会产生 CloudWatch 日志见解查询费用。

该estimate命令必须是查询中的最后一个命令。

> **Note**
>
> estimate返回的值是近似值，可能与运行查询时扫描的实际数据量不同。

语法

```
| estimate
```

示例

以下查询返回查询在不运行的情况下将扫描的数据量的估计值（以字节为单位）。

```
fields @timestamp, @message
| filter @message like /error/
| estimate
```

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
- 结构类型
- 日期时间函数
- 常见函数
- JSON 函数
- IP 地址字符串函数
- 字符串函数

算术运算符

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

操作	说明
`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	说明
=	Equal
!=	Not equal
<	Less than

运算符	说明
>	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	说明
`abs(a: number)`	number	绝对值
`ceil(a: number)`	数字	舍入到上限（大于 a 的值的最小整数）
`floor(a: number)`	数字	舍入到下限（小于 a 的值的最大整数）
`greatest(a: number, ...numbers: number[])`	数字	返回最大值
`least(a: number, ...numbers: number[])`	数字	返回最小值
`log(a: number)`	数字	自然对数
`round(a: number [, d: number])`	数字	对的值进行四舍五入a。使用一个参数，四舍五入到最接近的整数。如果有两个参数，则四舍五入到d小数位。
`sqrt(a: number)`	数字	平方根

操作	结果类型	说明
haversine(lat1: number, lon1: number, lat2: number, lon2: number)	数字	计算两个地理点之间的大圆距离，以千米为单位，由纬度和经度指定。
toNumber(fieldName: string)	数字	将字符串字段值转换为数值。
toInt(fieldName: string)	数字	将字段值转换为整数（32 位）。
toLong(fieldName: string)	数字	将字段值转换为长整数（64 位）。
toDouble(fieldName: string)	数字	将字段值转换为双精度浮点数。

结构类型

地图或列表是 L CloudWatch ogs Insights 中的一种结构类型，允许您访问和使用属性进行查询。

示例：获取映射或列表

使用 jsonParse 可以将 json 字符串字段解析为地图或列表。

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

示例：访问属性

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

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

使用方括号访问运算符（list[index]）可以检索列表中特定位置的项。

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

```
| filter json_message.users[1].action = "PutData"
```

当键名中出现特殊字符时，请用反引号（``）将特殊字符括起来。

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

示例：空结果

对于字符串、数字和日期时间函数，映射和列表会被视为空值。

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

将映射和列表与任何其他字段进行比较会得出 `false`。

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

日期时间函数

日期时间函数

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

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

例如，`10m` 为 10 分钟，`1h` 为 1 小时。

> **Note**
>
> 为日期时间函数使用最合适的时间单位。 CloudWatch 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 `s` 的任何请求，它将最大值限制为 60。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，请改用 bin(5m)。ms 的上限为 1000，s 和 m 的上限为 60，h 的上限为 24。

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

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

函数	结果类型	说明

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

函数	结果类型	说明
		• day d • week w • month mo mon • quarter q qtr • year y yr
datefloor(timestamp: Timestamp, period: Period)	Timestamp	将时间戳截断到指定的时间段。例如，datefloor(@timestamp, 1h) 将 @timestamp 的所有值截断至小时底部。
dateceil(timestamp : Timestamp, period: Period)	Timestamp	将时间戳向上舍入到指定的时间段，然后截断。例如，dateceil(@timestamp, 1h) 将 @timestamp 的所有值截断至小时顶部。
fromMillis(fieldName: number)	Timestamp	将输入字段解释为自 Unix epoch 以来的毫秒数并将其转换为时间戳。
toMillis(fieldName: Timestamp)	数字	将在命名字段中找到的时间戳转换为表示自 Unix epoch 以来毫秒数的数字。例如，toMillis(@timestamp) 将时间戳 2022-01-14T13:18:031.000-08:00 转换为 1642195111000 。

函数	结果类型	说明

now()	数字	返回查询处理开始的时间，以 epoch 秒为单位。此函数不接受任何参数。
		
		您可以使用它根据当前时间筛选查询结果。
		
		例如，以下查询将返回过去两小时内的所有 4xx 错误：
		
		`parse @message "Status Code: *;" as statusCode\n`
		`| filter statusCode >= 400 and statusCode <= 499 \n`
		`| filter toMillis(@timestamp) >= (now() * 1000 - 7200000)`
		
		以下示例会返回过去五小时内所有包含 error 或 failure 字样的日志条目
		
		`fields @timestamp, @message`
		`| filter @message like /(?i)(error|failure)/`
		`| filter toMillis(@timestamp) >= (now() * 1000 - 18000000)`

parseDate(fieldName: string, format: string [, timezone: string])	数字	使用 Java 模式将日期字符串解析为纪元毫秒。DateTimeFormatter 可选timezone参数指定用于解析的时区。 示例：`fields parseDate(field, "yyyy-MM-dd", "UTC") as epoch`
formatDate(timestamp: LogField, format: string [, timezone: string])	字符串	使用 strftime 风格的格式字符串格式化时间戳。也可用作别名strftime。可选timezone参数指定格式化的时区。 示例：`fields formatDate(@timestamp, "%Y-%m-%d", "UTC") as date`

> **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	说明
`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
`case(cond1, val1, cond2, val2, ..., [default])`	LogField	按顺序评估条件并返回第一个真实条件的值。如果没有条件为真且提供了默认值，则返回默认值。最多支持 10 个分支。
`if(condition: Boolean, trueValue: LogField, falseValue: LogField)`	LogField	评估一个条件，`trueValue` 如果条件为真或`falseValue` 不是，则返回。
`isNumeric(fieldName: LogField)`	布尔值	返回字段值`true`是否可以解析为数字。 示例：`filter isNumeric(@duration)`
`messageSize(fieldName: LogField)`	数字	返回字符串字段的字节长度。 示例：`fields messageSize(@message) as size`

函数	结果类型	说明
queryStartTime()	数字	以 epoch 毫秒为单位返回查询窗口的开始时间。
queryEndTime()	数字	以 epoch 毫秒为单位返回查询窗口的结束时间。
queryTimeRange()	数字	返回查询窗口的持续时间（以毫秒为单位）。

JSON 函数

json 函数

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

函数	结果类型	说明
jsonParse(fieldName: string)	映射 | 列表 | 空	当输入为 JSON 对象的字符串表示形式或 JSON 数组时，返回映射或列表。如果输入不是其中一种表示形式，则返回空值。
jsonStringify(fieldName: Map | List)	字符串	从映射或列表数据中返回 JSON 字符串。
jsonArraySize(fieldName: LogField)	数字	返回 JSON 数组字符串字段的元素数量。 示例：fields jsonArraySize(Operation) as arrayLen
jsonArrayContains(fieldName: LogField, value: string)	布尔值	如果字段中的 JSON 数组包含指定值，则返回 true (1)。对于无效的 JSON、非数组或空

函数	结果类型	说明
		数组，则返回 `false` (0)。使用区分大小写的比较。 示例：`filter jsonArrayContains(@roles, "admin")`

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	说明
`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。

函数	结果类型	说明
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。
ipv4ToNumber(fieldName: string)	数字	将 IPv4 地址字符串转换为其数字表示形式。
isPrivateIP(fieldName: string)	布尔值	true如果该 IP 地址在私有范围内，则返回 (RFC 1918)。
isPublicIP(fieldName: string)	布尔值	true如果该 IP 地址可以公开路由，则返回。
isReservedIP(fieldName: string)	布尔值	true如果该 IP 地址在保留范围内，则返回。

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	说明
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)	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

函数	结果类型	说明
ltrim(str: string, trimChars: string)		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("xyZxyfooxyxyZ","xyZ") 将返回 "foo"。
strlen(str: string)	数字	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。

函数	结果类型	说明
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。
regexReplace(fieldName: string, pattern: string, replacement: string)	字符串	将所有pattern与replacement 正则表达式匹配的子字符串替换为。使用 RE2 正则表达式语法。
strcontains(str: string, searchValue: string) strcontains(str: string, searchValue: string, caseInsensitive: boolean)	数字	如果 str 包含 searchValue ，则返回 1，否则返回 0。如果第三个参数设置为true，则匹配项不区分大小写。
startsWith(str: string, searchValue: string)	数字	如果以str开头则返回1searchValue ，否则返回0。

函数	结果类型	说明
`endsWith(str: string, searchValue: string)`	数字	如果`str`以结尾则返回`1searchValue` ，否则返回0。
`urlencode(str: string)`	字符串	URL-encodes 字符串。
`urldecode(str: string)`	字符串	URL-decodes 字符串。
`base64encode(str: string)`	字符串	Base64-encodes 字符串。
`base64decode(str: string)`	字符串	Base64-decodes 字符串。
`split(str: string, delimiter: string)`	array	按指定的分隔符分割字符串并返回子字符串数组。
`hexToAscii(value: string)`	字符串	将十六进制字符串转换为ASCII 文本。 示例：`fields hexToAscii("48656c6c6f") as text`
`hexToDec(value: string)`	数字	将十六进制字符串转换为十进制数。 示例：`fields hexToDec("0xff") as dec`
`decToHex(value: number)`	字符串	将十进制整数转换为带`0x`前缀的小写十六进制字符串。负数产生`-0x`前缀。 Non-integers被截断。 示例：`fields decToHex(255) as hex`

哈希函数

可以在`fields`和`filter`命令中使用哈希函数。

函数	结果类型	说明
`md5(fieldName: string)`	字符串	计算字符串值的 MD5 哈希值。
`sha256(fieldName: string)`	字符串	计算字符串值的 SHA-256 哈希值。

Time-series 函数

使用带有`stats`命令的时间序列函数来分析时间窗口内的指标并计算变化率。

函数	结果类型	说明
`rate(fieldName: NumericLogField, interval: Period)`	数字	计算数值字段的每个间隔变化率。
`countOverTime(fieldName: LogField)`	数字	计算每个时间 bin 的日志事件。与一起使用`by bin(interval)` 来设置窗口。
`sumOverTime(fieldName: NumericLogField)`	数字	对每个时间段的字段值求和。与一起使用`by bin(interval)` 来设置窗口。
`histogram(fieldName: NumericLo`	映射	将数值字段值存储到指定数量的等宽范围内，并返回分布。

函数	结果类型	说明
gField, buckets: number)		

offset

offset在stats ... by bin()子句末尾使用，将时间序列数据桶按指定的持续时间移动。这样可以进行时移比较，例如将当前指标与前一小时或一天的同期进行比较。

语法

```
stats <aggregation> by bin(<period>) offset <duration>
```

示例

```
stats count(*) by bin(5m) offset 1h
```

```
stats avg(latency) by bin(1m) offset 1d
```

包含特殊字符的字段

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

对于以其余名称开头@并在其余名称中包含特殊字符的系统字段，请将@符号放在反引号外并仅包含字段名称部分。例如，资源标签字段使用该模式@aws.tag.<tagKey>。如果标签密钥包含冒号等特殊字符，则必须使用以下语法：

```
@`aws.tag.aws:cloudformation:stack-name`
```

以下示例显示了使用此语法的过滤器查询：

```
filter @`aws.tag.aws:cloudformation:stack-name` = "my-stack"
```

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
```

开始使用 Logs Insights QL：查询教程

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

主题

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

教程：运行和修改示例查询

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

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

3. 在选择日志组下拉菜单中，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**
   >
   > 仅标准日志类中的日志组支持已发现字段。有关日志类的更多信息，请参阅 日志类。

4. （可选）使用时间间隔选择器选择要查询的时间段。

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

5. 选择 Run（运行）以查看结果。

   在本教程中，结果包括 20 个最近添加的日志事件。

   CloudWatch 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。

Logs Insights QL 包含强大的查询能力，包括若干命令和对于正则表达式、数学和统计运算的支持。有关更多信息，请参阅 CloudWatch 日志见解语言查询语法。

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. 确认已选中 Logs Insights QL 选项卡。
4. 在选择日志组下拉菜单中，选择要查询的一个或多个日志组。

   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。
5. 在查询编辑器中删除原定设置查询，然后输入以下命令：

   ```
   stats count(*) by fieldName
   ```
6. *fieldName*替换为 “字段” 菜单中已发现的字段。

   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。
7. 选择 Run（运行）以查看查询结果。

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

教程：运行生成按日志字段分组的可视化的查询

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

运行查询进行可视化

1. 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. 确认已选中 Logs Insights QL 选项卡。
4. 在选择日志组下拉菜单中，选择要查询的一个或多个日志组。

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

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

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

6. 选择可视化选项卡。

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。

查询示例

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 日志见解语言查询语法。

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
- 查询资源标签
- 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

查找每项服务的日志条目数、事件类型和 AWS 区域。

```
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 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 countDistinct(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
```

查询资源标签

按资源标签值筛选日志。

```
filter @aws.tag.env = "production"
```

按包含特殊字符的标签密钥进行筛选。

```
filter @`aws.tag.aws:cloudformation:stack-name` = "my-stack"
```

获取按标签值分组的日志事件计数。

```
stats count(*) by @aws.tag.team
```

解析命令的示例

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

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

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

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

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

```
FIELDS @message
```

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

与之前的时间范围进行比较（差异）

您可以使用 L CloudWatch ogs Insights 和 Logs Insights QL 来比较日志事件在一段时间内的变化。您可以将最近时间范围内提取的日志事件与紧接的前一个时间段的日志进行比较。或者，您可以与类似的过去时间段进行比较。这可以帮助您发现日志中的错误是最近引入的还是已经发生的，并可以帮助您发现其他趋势。

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

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

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

> **Note**
>
> 使用比较的查询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，依次选择日志、Logs Insights。

   查询框中会出现默认查询。
3. 确认已选中 Logs Insights QL 选项卡。
4. 保留默认查询或输入其他查询。

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

   为了使查询从比较时间段获取数据，将在您的查询中附加 `diff` 命令。
9. 选择模式选项卡以查看结果。

   该表将显示以下信息：

   - 每个模式，模式的可变部分替换为动态令牌符号 <*string*-*number*>。*string*是对令牌所代表的数据类型的描述。*number*显示了与其他动态代币相比，该代币在模式中的显示位置。有关更多信息，请参阅 模式分析。
   - 事件计数是原始的、较新的时间段内具有该模式的日志事件的数量。
   - 差异事件计数是当前时间段与比较时间段内匹配的日志事件数之间的差异。正差异表示当前时间段内此类事件较多。
   - 差异描述简要总结了当前时间段和比较时间段之间的模式变化。
   - 严重性类型是具有此模式的日志事件的可能严重性，基于在日志事件中发现的字词（例如 FATAL、ERROR 和 WARN）。
10. 要进一步检查列表中的某个模式，请在检查列中选择其中一个模式的图标。

    将出现模式检查窗格并显示以下内容：

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

      > **Note**
      >
      > 每个令牌最多可捕获 10 个令牌值。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与您正在检查的模式几乎同时频繁出现的其他模式。例如，如果 ERROR 消息的模式通常伴随另一个标记为 INFO 并带有附加详细信息的日志事件，则此处显示该模式。

在图形中可视化日志数据

您可以使用条形图、折线图和堆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。

OpenSearch 管道处理语言 (PPL)

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

使用 AWS CLI 或 API 创建查询时，SOURCE在 PPL 查询中包含是指定要包含在查询中的日志组、字段索引和数据源的有用方法。只有 AWS CLI 和 API 支持该SOURCE命令， CloudWatch 控制台不支持该命令。使用 CloudWatch控制台启动查询时，您可以使用控制台界面来指定日志组以及数据源的名称和类型。

使用 aws:fieldIndex 可以强制查询仅扫描在查询中指定的字段上编制索引的日志组，从而返回已编制索引的数据 根据filterIndex命令中指定的字段，将自动选择相关的日志组。这会跳过没有任何包含查询中指定字段的日志事件的日志组，而只扫描与该字段索引的查询中指定的值相匹配的日志组，从而减少扫描量。aws:fieldIndex用于在源命令中指定字段名称以及字段名称和值，以便仅查询包含指定字段和值的索引数据。有关更多信息，请参阅 创建字段索引以提高查询性能并减少扫描量。

您可以使用 OpenSearch PPL 查询标准日志类中的日志组。

> **Note**
>
> 有关 CloudWatch 日志中支持的所有 OpenSearch PPL 查询命令的信息以及语法和限制的详细信息，请参阅《 OpenSearch 服务开发者指南》中的 “支持的 PPL 命令”。
> 有关您可以使用的其他查询语言的信息，请参阅 L CloudWatch ogs Insights、Serv OpenSearch ice SQL 和 CloudWatch Metrics Insights

命令或函数	示例查询	说明
fields	fields field1, field2	显示一组需要投影的字段。

命令或函数	示例查询	说明
join	`LEFT JOIN left=l, right=r on l.id = r.id `join_right_lg` | fields l.field_1, r.field_2`	将两个数据集联接在一起。
where	`where field1="success" | where field2 != "i-023fe0a90929d8822" | fields field3, field4, field5,field6 | head 1000`	根据指定的条件筛选数据。
aws:fieldIndex	`source = [`aws:fieldIndex`="region", `region` = "us-west-2"] | where status = 200 | head 10`	通过强制查询仅扫描在查询中指定的字段上建立索引的日志组，仅返回已编入索引的数据。
stats	`stats count(), count(field1), min(field1), max(field1), avg(field1) by field2 | head 1000`	执行聚合和计算
parse	`parse field1 ".*/(?<field2>[^/]+$)" | where field2 = "requestId" | fields field1, field2 | head 1000`	从字符串中提取正则表达式（regex）模式，并显示提取出的模式。提取出的模式可进一步用于创建新字段或筛选数据。
sort	`stats count(), count(field1), min(field1) as field1Alias, max(`field1`), avg(`field1`) by field2 | sort -field1Alias | head 1000`	按字段名称对显示的结果进行排序。使用 sor FieldName t-按降序排序。

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

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

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

命令或函数	示例查询	说明
Date-Time 函数	`eval newDate = ADDDATE(DATE('2020-08-26'), 1) | fields newDate`	Built-in 用于在 PPL 查询中处理和转换日期和时间戳数据的函数。例如，date_add、date_format、datediff、date-sub、timestampadd、timestampdiff、current_timezone、utc_timestamp 和 current_date。
条件函数	`eval field2 = isnull(field1) | fields field2, field1, field3`	Built-in 用于检查特定字段条件并有条件地评估表达式的函数。例如，如果 field1 为空值，则返回 field2。

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

查询范围

使用 AWS CLI 或 API 创建查询时，在查询中包含 SOURCE 是指定要包含在查询中的日志组的有用方法。只有 AWS CLI 和 API 支持 SOURCE 命令， CloudWatch控制台不支持。使用 CloudWatch 控制台启动查询时，您可以使用控制台界面来指定日志组以及数据源的名称和类型。

PPL 的 source 命令现在支持多种指定它们的方式：

1. 日志组
2. 字段索引-新增
3. 数据源和类型-新建

日志组

当客户知道需要搜索哪些确切的日志组时，可以使用日志组来源选择

```
source = [lg:`/aws/lambda/my-function`] | where status = 200 | head 10
```

字段索引

当筛选器针对已编制索引的字段时，基于字段索引的源选择将结果限制为仅包含索引的数据，从而减少了查询的数据量。根据filterIndex命令中指定的字段，将自动选择相关的日志组。有关字段索引及其创建方法的更多信息，请参阅创建字段索引以提高查询性能和减少扫描量。

使用 aws:fieldIndex 可以强制查询仅扫描在查询中指定的字段上编制索引的日志组，从而返回已编制索引的数据 对于已在此字段上编制索引的日志组，它会进一步优化查询，跳过没有任何日志事件包含索引字段的查询中指定的字段的日志组。这样可以进一步减少扫描量，因为它会尝试仅扫描这些日志组中与此字段索引的查询中指定的值匹配的日志事件。有关字段索引及其创建方法的更多信息，请参阅创建字段索引以提高查询性能和减少扫描量。

在 PPL 中，aws:fieldIndex用于指定应将哪些键值对视为索引。语法如下

```
source = [`aws:fieldIndex`="region", `region` = "us-west-2"] | where status = 200 |
 head 10
```

其中，

1. `aws:fieldIndex`="region"将区域标识为字段索引。
   a. 注意：客户可以使用 IN 来指定多个索引，而不是 =（以下示例）
2. `region`="us-west-2"标识要应用的过滤条件
   a. 注意：客户可以使用 IN 来指定多个值，而不是 =（以下示例）

客户可以指定多个字段索引，如下所示

```
source = [`aws:fieldIndex` IN ("status", "region"), `status` = 200, `region` IN ("us-
west-2", "us-east-1")] | head 10
```

数据源和类型

当客户知道需要查询哪些确切的数据源时，可以使用基于数据源和类型的源选择。此查询是在一个或多个包含指定数据源和类型的日志组上执行的。

```
source = [ds:`data_source.type`] | where status = 200 | head 10
```

支持用于数据源查询的 PPL

要支持在 PPL 中查询数据源的用例，您可以使用动态源选择器子句。使用此语法，您可以通过在搜索命令中指定数据源来查询数据源。您最多可以指定 10 个数据源。

语法

```
source=[ds:`DataSource1.Type1`, ds:`DataSource2.Type2`, ...ds:`DataSourcen.Typen`]
```

示例查询

```
search source=[ds:`DataSource1.Type1`, ds:`DataSource2.Type2`] | fields field1, field2
```

组合示例

客户可以按任意顺序指定所有来源选择运算符，结果将是所有应用条件的交集。

例如，/aws/lambda/my-function-1 可能包含多个数据源和类型，包括各种各样的索引，当运行以下查询时，返回的结果将只有源和类型的事件， DataSource1.Type1 并且符合 “status” = 200 的标准。

```
search source=[
    ds:`DataSource1.Type1`,
    lg:`/aws/lambda/my-function-1`,
    `aws:fieldIndex` IN ("status"), `status` = 200
]
```

限制

当您使用 OpenSearch PPL 在 Lo CloudWatch gs Insights 中进行查询时，以下限制适用。

- 不能对数据源查询使用联接或子查询命令。

OpenSearch 结构化查询语言 (SQL)

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

使用 `filterIndex` 可以强制查询仅扫描在查询中指定的字段上编制索引的日志组，从而返回已编制索引的数据 跳过没有任何包含查询中指定字段的日志事件的日志组，而只扫描与该字段索引查询中指定的值相匹配的日志组，从而减少扫描量。`filterIndex`用于指定字段名称以及字段名称和值，以便仅查询包含指定字段和值的索引数据。

您可以使用 OpenSearch SQL 查询标准日志类中的日志组。SQL 还支持使用数据源名称和数据源类型进行查询。

> **Note**
>
> 下表列出了 CloudWatch 日志中支持的 SQL 命令和函数。有关包括语法在内的所有 OpenSearch SQL 命令的信息，请参阅《 OpenSearch 服务开发人员指南》中支持的 SQL 命令。
> 有关您可以使用的其他查询语言的信息，请参阅 L CloudWatch ogs Insights、Ser OpenSearch vice PPL 和CloudWatch Metrics Insights。

支持的 SQL 命令

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

命令或函数	示例查询	说明
SELECT	`SELECT `@message`, Operation FROM `LogGroupA``	显示预测值。

命令或函数	示例查询	说明
FROM	`SELECT `@message`, Operation FROM `LogGroupA``	Built-in 子句，它指定要从中检索数据的源表或视图，支持各种类型的联接和子查询。
WHERE	`SELECT * FROM `LogGroupA` WHERE Operation = 'x'`	根据提供的字段条件筛选日志事件。
filterIndex	`SELECT * FROM `filterIndex('region' = 'us-east-1')` WHERE status = 200 LIMIT 10;`	通过强制查询仅扫描在查询中指定的字段上建立索引的日志组，仅返回已编入索引的数据。
GROUP BY	`SELECT `@logStream`, COUNT(*) as log_count FROM `LogGroupA` GROUP BY `@logStream``	根据类别对日志事件进行分组，并根据统计数据计算平均值。
HAVING	`SELECT `@logStream`, COUNT(*) as log_count FROM `LogGroupA` GROUP BY `@logStream` HAVING log_count > 100`	根据分组条件筛选结果。
ORDER BY	`SELECT * FROM `LogGroupA` ORDER BY `@timestamp` DESC`	根据 ORDER BY 子句中的字段对结果进行排序。您可以按降序或升序进行排序。

命令或函数	示例查询	说明
JOIN	`SELECT A.`@message`, B.`@timestamp`FROM `LogGroupA` as A INNER JOIN `LogGroupB` as B ON A.`requestId` = B.`requestId``	根据共同字段合并两个表的结果。必须指定内部联接或左外联接
LIMIT	`Select * from `LogGroupA` limit 10`	将显示的查询结果限制为前 N 行。
字符串函数	`SELECT upper(Operation) , lower(Operation), Operation FROM `LogGroupA``	Built-in SQL 中的函数，可以在 SQL 查询中操作和转换字符串和文本数据。例如，转换大小写、合并字符串、提取部分内容以及清理文本。
日期函数	`SELECT current_date() as today, date_add(current_date(), 30) as thirty_days_later, last_day(current_date()) as month_end FROM `LogGroupA``	Built-in 用于在 SQL 查询中处理和转换日期和时间戳数据的函数。例如，date_add、date_format、datediff 和 current_date。

命令或函数	示例查询	说明
条件函数	`SELECT Operation, IF(Error > 0, 'High', 'Low') as error_category FROM `LogGroup A`;`	Built-in 基于指定条件执行操作或有条件地评估表达式的函数。例如，CASE 和 IF。
聚合函数	`SELECT AVG(bytes) as bytesWritten FROM `LogGroupA``	Built-in 对多行执行计算以生成单个汇总值的函数。例如，SUM、COUNT、AVG、MAX 和 MIN。
JSON 函数	`SELECT get_json_object(json_column, '$.name') as name FROM `LogGroupA``	Built-in 用于解析、提取、修改和查询 SQL 查询中 JSON-formatted 数据的函数（例如，from_json、to_json、get_json_object、json_tuple），允许操作数据集中的 JSON 结构。

命令或函数	示例查询	说明
数组函数	`SELECT scores, size(scores) as length, array_contains(scores, 90) as has_90 FROM `LogGroupA`;`	Built-in 用于在 SQL 查询中处理数组类型列的函数，允许诸如访问、修改和分析数组数据（例如，大小、爆炸、array_contains）之类的操作。
窗口函数	`SELECT field1, field2, RANK() OVER (ORDER BY field2 DESC) as field2Rank FROM `LogGroupA`;`	Built-in 对与当前行（窗口）相关的一组指定行执行计算的函数，可实现排名、运行总数和移动平均值等操作。例如，ROW_NUMBER、RANK、LAG 和 LEAD
转换函数	`SELECT CAST('123' AS INT) as converted_number, CAST(123 AS STRING) as converted_string FROM `LogGroupA``	Built-in 用于在 SQL 查询中将数据从一种类型转换为另一种类型、支持数据类型转换和格式转换的函数。例如，CAST、TO_DATE、TO_TIMESTAMP 和 BINARY。

命令或函数	示例查询	说明
谓词函数	`SELECT scores, size(scores) as length, array_contains(scores, 90) as has_90 FROM `LogGroupA`;`	Built-in 根据指定条件或模式评估条件并返回布尔值 (true/false) 的函数。例如，IN、LIKE、BETWEEN、IS NULL 和 EXISTS。
选择多个日志组	`SELECT lg1.field1, lg1.field2 from `logGroups(logGroupIdentifier: ['LogGroup1', 'LogGroup2'])` as lg1 where lg1.field3= "Success"`	使您能够在 SELECT 语句中指定多个日志组
选择多个数据源	`SELECT ds1.field1, ds1.field2 from `dataSource(['DataSource1', 'DataSource2'])` as ds1 where ds1.field3= "Success"`	使您能够在 SELECT 语句中指定多个数据源

多日志组查询支持的 SQL

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

语法：

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

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

示例查询：

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

查询日志时，不支持在语FROM句后面使用以下涉及多个 CloudWatch 日志组的语法。

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

支持用于数据源查询的 SQL

要支持在 SQL 中查询数据源的用例，可以使用 dataSource 命令。使用此语法，您可以通过在FROM命令中指定数据源来查询数据源。您最多可以指定 10 个数据源。

语法

```
`dataSource(
    ['DataSource1', 'DataSource2', ...'DataSourcen']
)`
```

查询示例

```
SELECT DS1.Column1, DS1.Column2 from `dataSource(
    ['DataSource1', 'DataSource2']
)` as DS1 WHERE DS1.Column1 = 'ABC'
```

查询范围

在 AWS CLI 和 API 中，您可以使用日志组、数据源和类型以及字段索引来指定要查询的日志。

日志组

当客户知道需要搜索哪些确切的日志组时，可以使用日志组来源选择

```
SELECT * FROM `logGroups(logGroupIdentifier: ['/aws/lambda/my-function'])`;
```

数据源和类型

客户可以使用数据源名称和数据源类型查询日志。

当客户知道需要查询哪些确切的数据源时，可以使用基于数据源和类型的源选择。此查询是在一个或多个包含指定数据源和类型的日志组上执行的。

要支持在 SQL 中查询数据源的用例，可以使用 dataSource 命令。使用此语法，您可以通过在 FROM 命令中指定数据源来查询数据源。您最多可以指定 10 个数据源。

语法：

```
`dataSource(
    ['DataSource1.Type1', 'DataSource2.Type2', ...'DataSourcen.Typen']
)`
```

示例查询：

```
SELECT DS1.Column1, DS1.Column2 from `dataSource(
    ['DataSource1.Type1', 'DataSource2.Type2']
)` as DS1 WHERE DS1.Column1 = 'ABC'
```

有关按数据源进行查询的更多信息，请参阅使用分面对日志进行分组和浏览。

组合示例

客户可以按任意顺序在反引号内指定所有源选择运算符，结果将基于所应用的所有条件的交集。

例如，/aws/lambda/my-function-1 可能包含多个数据源和类型，包括各种各样的索引，当运行以下查询时，返回的结果将只有源和类型的事件， DataSource1.Type1 并且符合 “status” = 200 的标准。

```
SELECT * FROM `
   logGroups(logGroupIdentifier: ['/aws/lambda/my-function'])
   filterIndex('status' = 200)
   dataSource(['DataSource1.Type1'])
`;
```

字段索引

当您的筛选器针对索引字段时，字段 Index-based 源选择会自动识别相关的日志组，从而减少扫描量和查询运行时间。

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

中与此字段索引的查询中指定的值匹配的日志事件。有关字段索引及其创建方法的更多信息，请参阅[创建字段索引以提高查询性能和减少扫描量]。

在 SQL 中，filterIndex 用于指定应将哪些键值对视为索引。语法如下

```
SELECT * FROM `filterIndex('region' = 'us-east-1')`;
```

其中，

1. FilterIndex (...) 指定，将其中的键值视为字段索引。每个键值对用逗号分隔（以下示例）
2. 'region'='us-east-1'指定了要应用的实际条件
   a. 注意：客户可以使用 IN 来指定多个值，而不是 =（以下示例）

使用多个 FilterIndex 会使用 “AND” 组合条件。在示例中，将查询与 status=200 和 us-east-1 或 us-west-2 中的区域匹配的日志。

```
SELECT * FROM `filterIndex('status' = 200, 'region' IN ['us-east-1', 'us-west-2'])`;
```

限制

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

- SELECT 语句中只能包含一个 JOIN。
- 不能将 JOIN 或子查询用于数据源查询。
- 仅支持一层嵌套子查询。
- 不支持用分号（;）分隔的多语句查询。
- 不支持包含仅大小写不同的相同字段名（例如 field1 和 FIELD1）的查询。

  例如，不支持以下查询：

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

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

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

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

例如，不支持此查询：

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

支持此查询：

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

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

  包含简单字段的查询示例：

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

  附加反引号的类似查询：

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

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

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、Serv OpenSearch ice PPL、Serv OpenSearch ice SQL 和CloudWatch Metrics Insights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。自然语言功能可根据您输入的提示生成查询，并逐行说明查询的工作原理。您也可以更新查询以进一步调查数据。

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

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

要生成具有此功能的 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，您必须使用、CloudWatchLogsFullAccessCloudWatchLogsReadOnlyAccessAdministratorAccess、或ReadOnlyAccessIAM 策略登录，或者拥有相应`cloudwatch:GenerateQuery`权限。

示例查询

本节中的示例描述了如何使用自然语言功能生成和更新查询。

> **Note**
>
> 有关 CloudWatch Logs Insights 查询编辑器和语法的更多信息，请参阅 CloudWatch Logs Insights 查询语法。

示例：生成自然语言查询

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

提示

以下提示示例指示该功能搜索速度最慢的 10 个 Lambda 函数调用。

```
Find the 10 slowest requests
```

Query

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

```
# Find the 10 slowest requests
fields @timestamp, @message, @duration
| sort @duration desc
| limit 10
```

```
# This query retrieves the timestamp, message and duration fields from the logs and
 sorts them in descending order by duration to find the 10 slowest requests.
```

> **Note**
>
> 要关闭提示和有关查询工作原理的说明显示，请使用编辑器中的齿轮图标。

提示

要生成 S OpenSearch QL 查询，请选择 OpenSearch SQL 选项卡，然后打开查询生成器提示框以输入您的自然语言提示。以下是使用自然语言功能生成 OpenSearch SQL 查询的提示示例。

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

Query

下面是该提示生成的 SQL 查询，可用于查找每小时汇总的错误和异常数量：

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

提示

要生成 OpenSearch PPL 查询，请选择 OpenSearch PPL 选项卡，然后打开查询生成器提示框以输入您的自然语言提示。以下是使用自然语言功能生成 OpenSearch PPL 查询的提示示例。

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

Query

下面是该提示生成的 PPL 查询，您可以使用它来查找日志中的唯一异常消息：

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

示例：更新自然语言查询

您可以通过编辑初始提示，然后选择更新查询，来更新查询。

更新版提示

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

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

更新版查询

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

```
# Show top 20 slowest requests instead and display requestId as a column
fields @timestamp, @message, @requestId, @duration
| sort @duration desc
| limit 20
# This query modifies the original query by replacing the @message field with the
 @requestId field and changing the limit from 10 to 20 to return the top 20 log events
 by duration instead of the top 10.
```

选择不使用您的数据来改进服务

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。

支持的日志和发现的字段

CloudWatch Logs Insights 支持不同的日志类型。对于发送到 Amazon 日志中标准类日志组的每份 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。
- `@entity` 包含探索相关遥测功能的与实体相关的扁平化 JSON。

  例如，此 JSON 可以代表一个实体。

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

  对于此实体，提取的系统字段如下所示：

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

> **Note**
>
> 仅标准日志类中的日志组支持字段发现。有关日志类的更多信息，请参阅 日志类。

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

对于许多日志类型， CloudWatch 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 日志见解语言查询语法。

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`。

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

日志类型	发现的日志字段
Amazon VPC 流日志	`@timestamp` , `@logStream` , `@message`, `accountId` , `endTime`, `interfaceId` , `logStatus` , `startTime` , `version`, `action`, `bytes`, `dstAddr`, `dstPort`, `packets`, `protocol`, `srcAddr`, `srcPort`
Route 53 日志	`@timestamp` , `@logStream` , `@message`, `edgeLocation` , `ednsClientSubnet` , `hostZoneId` , `protocol`, `queryName` , `queryTimestamp` , `queryType` , `resolverIp` , `responseCode` , `version`
Lambda 日志	`@timestamp` , `@logStream` , `@message`, `@requestId` , `@duration`, `@billedDuration` , `@type`, `@maxMemoryUsed` , `@memorySize` 如果 Lambda 日志行包含 X-Ray 跟踪 ID，则它还包括以下字段：`@xrayTraceId` 和。`@xraySegmentId`

日志类型	发现的日志字段
	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。

点符号和 JSON-encoded 字符串字段

点符号仅遍历在采集时存储为结构嵌套 JSON 对象的字段。如果字段的值是 JSON-encoded 字符串（其内容恰好是有效 JSON 的字符串），则点表示法将其视为不透明的叶子值，并且不访问其中的子字段。

这种情况通常发生在：

- 由 OCSF 管道转换的日志（例如`api.request.data`，按`json_t`照 OCSF 架构键入）
- 任何在摄取之前将可变结构有效载荷序列化为字符串的日志源

例如，如果`api.request.data`包含`{"startTime":1234,"endTime":5678}`为字符串值，则不`api.request.data.startTime`返回任何结果，因为 CloudWatch Logs Insights 会将整个字符串视为单个叶值而不是嵌套对象。

要访问 JSON-encoded 字符串中的子字段，请使用将字符串`jsonParse`转换为可遍历地图。

示例：使用 jsonParse 查询字符串字段 JSON-encoded

```
fields jsonParse(api.request.data).startTime as startTime
```

此查询用于`jsonParse`将 JSON-encoded 字符串解析`api.request.data`为地图，然后使用点符号访问`startTime`子字段。有关的更多信息`jsonParse`，请参阅结构类型。

> **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。

创建字段索引以提高查询性能并减少扫描量

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

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

您还可以使用字段索引创建对更多日志组的高效查询。当您在查询中使用 `filterIndex` 命令而不是 `filter` 命令时，查询将针对具有字段索引的日志事件的选定日志组运行。这些查询最多可以扫描 10,000 个日志组，您可以通过指定最多五个日志组名称前缀进行选择。如果这是 CloudWatch 跨账户可观察性的监控账户，则可以选择所有源账户或指定单个源账户来选择日志组。

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

仅 JSON 和服务日志的结构化日志格式支持字段索引。

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

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

CloudWatch 日志还为某些数据源名称和类型组合提供默认字段索引。默认字段索引自动适用于以下数据源名称和类型组合：

数据源名称和类型	默认字段索引
`amazon_vpc.flow`	`action` `logStatus`

数据源名称和类型	默认字段索引
	region flowDirection type
amazon_route53.resolver_query	query_type transport rcode
aws_waf.access	action httpRequest.country
aws_cloudtrail.data aws_cloudtrail.management	eventSource eventName awsRegion userAgent errorCode eventType managementEvent readOnly eventCategory requestId

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

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

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

本节的其余主题将介绍自动编制索引的字段以及如何创建字段索引。有关如何在查询中引用字段索引的信息，请参阅 filterIndex 和 筛选。

主题

- 自动索引字段
- 字段索引语法和配额
- 创建账户级字段索引策略
- 创建日志组级字段索引策略
- 创建查询时的日志组选择选项
- 删除字段索引策略的影响

自动索引字段

CloudWatch 日志会根据字段最近在使用=或IN运算符的 CloudWatch Logs Insights 相等过滤器中的使用情况自动索引字段。这些索引是默认字段索引和您在策略中配置的字段索引的补充。您无需为自动编制索引的字段配置策略。

CloudWatch 日志会根据最近的查询活动更新一组自动编制索引的字段。当 CloudWatch Logs 从集合中移除某个字段时，它将停止为该字段新摄取的事件编制索引。 CloudWatch 日志管理自动编制索引的字段，并将其保留 30 天。要将字段永久编入索引，请将其添加到账户级别或日志组级别的字段索引策略中。

> **⚠ 对于自动编制索引的字段，使用过滤器而不是 filterIndex**
>
> 我们建议您使用`filter`而不是自动编制索引`filterIndex`的字段。 CloudWatch 日志可以根据查询模式更新或删除这些字段。 CloudWatch 日志仅将其保留 30 天。该`filterIndex`命令仅返回索引数据。由于 CloudWatch Logs 会根据查询活动更新集合，因此`filterIndex`查询不会搜索在 Log CloudWatch s 选择该字段之前或 Logs 停止为该字段编制索引之后 CloudWatch 提取的事件。如果您使用`filterIndex`，请将该字段添加到字段索引策略中，这样无论查询活动如何，都要对其进行索引。有关更多信息，请参阅 filterIndex 与 filter 之比较。

日志组的 “字段索引” 选项卡列出了自动编制索引的字段。

您也可以使用该DescribeFieldIndexes操作列出自动编制索引的字段。包含AUTO在indexCategories请求参数中。AUTO对于自动编制索引indexCategory的字段，响应设置为。省略的请求indexCategories不会返回自动编入索引的字段。

将自动编制索引的字段添加到日志组级别的索引策略中

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，依次选择日志和日志组。
3. 选择日志组的名称。
4. 选择 “字段索引” 选项卡，然后选择 “管理字段索引”。
5. 在管理日志组级别的字段索引页面上，查看策略中的字段。根据需要添加或删除字段，然后选择 “保存”。

字段索引语法和配额

您可以通过创建字段索引策略来创建字段索引。您可以创建适用于整个账户的账户级索引策略，也可以创建仅适用于单个日志组的策略。对于账户范围内的索引策略，您可以设置适用于账户中的所有日志组的索引策略。您还可以创建适用于账户中的部分日志组的账户级索引策略（按日志组名称的前缀选择）。如果在同一个账户中有多个账户级策略，则这些策略的日志组名称前缀不能重叠。同样，您可以创建适用于特定数据源名称和类型组合的账户级索引策略。每个数据源名称和类型组合只能创建一个账户策略。

日志组级别的字段索引策略优先于账户级字段索引策略：这些策略适用于整个日志组（例如，没有选择标准或基于日志组名称前缀的选择标准的账户级策略）。 Account-level 除了与整个日志组匹配的策略外，还将应用在日志事件级别匹配的策略（例如，针对给定的数据源名称和类型组合）。如果您创建日志组级别的索引策略，则该日志组不使用与日志组级别匹配的账户级策略。

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

您最多可以有 40 个账户级别的索引策略，其中 20 个可以使用日志组名称前缀选择标准，20 个可以使用基于数据源的选择标准。如果您有多个账户级索引策略，并且它们按日志组名称前缀进行筛选，则其中任何两个策略都不能使用相同或重叠的日志组名称前缀。例如，如果您有一个策略筛选出以 `my-log` 开头的日志组，则不能再有一个字段索引策略筛选出以 `my-logpprod` 或 `my-logging` 开头的日志组。同样，如果您有多个账户级索引策略筛选为数据源名称和类型组合，则其中任何两个策略都不能

使用相同的数据源名称和类型。例如，如果您将一个策略筛选为数据源名称amazon_vpc和数据源类型，flow则无法使用此组合创建另一个策略。

如果您的账户级索引策略没有名称前缀且适用于所有日志组，则无法创建其他带有日志组名称前缀筛选器的账户级索引策略；您可以创建使用数据源名称和类型筛选器的账户级索引策略。

每个索引策略都有以下配额和限制：

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

对于具有基于数据源名称和类型的选择标准的账户级索引策略，还有一个额外的限制：所有字段都必须是原始数据类型，只有结构支持嵌套基元。

生成的字段和保留字段

CloudWatch Logs Insights 会在每个日志事件中自动生成系统字段。这些生成的字段以 @ 为前缀。有关生成的字段的更多信息，请参阅支持的日志和发现的字段。

在这些生成的字段中，以下字段支持用作字段索引：

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

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

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

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

CloudWatch 日志还为某些数据源名称和类型组合提供默认字段索引。默认字段索引自动适用于以下数据源名称和类型组合：

数据源名称和类型	默认字段索引
amazon_vpc.flow	action logStatus region flowDirection type
amazon_route53.resolver_query	query_type transport rcode

数据源名称和类型	默认字段索引
aws_waf.access	action httpRequest.country
aws_cloudtrail.data aws_cloudtrail.management	eventSource eventName awsRegion userAgent errorCode eventType managementEvent readOnly eventCategory requestId

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

JSON 日志中的子字段和数组字段

您可以对 JSON 日志中的嵌套子字段或数组字段编制索引。

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

```
{
    "eventVersion": "1.0",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EXAMPLE_PRINCIPAL_ID",
        "arn": "arn: aws: iam: : 123456789012: user/Alice",
        "accessKeyId": "11112222",
        "accountId": "123456789012",
```

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

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

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

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

然后，此字段的查询可能如下所示：

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

创建账户级字段索引策略

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

创建账户级字段索引策略

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在左侧导航窗格上，选择设置，然后选择日志选项卡。
3. 在账户级索引策略部分，选择管理。
4. 选择创建索引策略。
5. 在策略名称中，输入新策略的名称。
6. 对于 “选择策略范围”，请执行以下任一操作：
   - 选择所有标准日志组，将索引策略应用于账户中的所有标准类日志组。
   - 选择 “按前缀匹配的日志组”，将策略应用于名称均以相同字符串开头的日志组子集。然后，在输入前缀名称中输入这些日志组的前缀。

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

     选择 “按数据源记录数据”，将策略应用于特定的数据源名称和类型组合。然后，您可以从下拉菜单中选择数据源和数据类型。

     选择数据源名称和类型后，您可以选择获取字段，在配置字段索引和分面部分填充相关信息，例如可用字段、包含的日志组以及默认和自定义字段索引。
7. 对于自定义索引字段配置，选择添加字段路径以输入要编制索引的第一个字段。

   然后输入用作字段名称值的字符串，或者从下拉菜单中选择一个字段。这必须与日志事件中出现的内容的大小写完全匹配。例如，如果您的日志事件包含 `requestId`，则必须在此处输入 `requestId`。`RequestId`、`requestID` 和 `request Id` 不匹配。

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

   您还可以为 L CloudWatch ogs 自动生成的`@ingestionTime`和`@logStream`字段创建索引。如果这样做，则在指定它们时无需添加额外的 @。
8. （可选）除了指定字段路径外，您还可以选择 “设置为分面”，将字段创建为分面。

9. 重复上一步，最多可添加 20 个字段索引。
10. 完成后，选择创建。

创建日志组级字段索引策略

请按照本节中的步骤创建适用于单个日志组的字段索引策略。

创建日志组级字段索引策略

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

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

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

   您还可以为 L CloudWatch ogs 自动生成的`@ingestionTime`和`@logStream`字段创建索引。如果这样做，则在指定它们时需要添加额外的 @。
7. （可选）除了指定字段路径外，您还可以选择 “设置为分面”，将字段创建为分面。
8. 重复上一步，最多可添加 20 个字段索引。
9. 完成后，选择 Save。

创建查询时的日志组选择选项

本节介绍了选择要包含在查询中的日志组的各种方法。

在控制台中选择用于查询的日志组

1. 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、Logs Insights。
3. 选择要用于此查询的查询语言。您可以选择：日志 Insights QL、OpenSearchPPL 或 OpenSearch SQL。
4. 可通过三种方式选择要用于查询的日志组：
   - 使用日志组名称框。这是默认选择方法。使用此方法，您最多可以输入 50 个日志组名称。如果这是 CloudWatch跨账户可观察性的监控账户，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。
   - 使用日志组条件部分。在此部分，您可以根据日志组名称的前缀选择日志组。一个查询最多可以包含五个前缀。名称中包含这些前缀的日志组将被选中。或者，所有日志组选项会选择账户中的所有日志组。
   - 如果这是 CloudWatch 跨账户可观察性的监控账户，则可以在账户下拉菜单中选择所有账户，从所有关联账户中选择日志组。或者，您可以单独选择要包含在此查询中的账户。

   如果您的选择与超过 10,000 个日志组匹配，您将看到一个错误，提示您缩小选择范围。
5. 查询的默认日志类为标准。您可以使用日志类将其更改为不频繁访问。

使用 AWS CLI

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

删除字段索引策略的影响

如果删除已生效一段时间的字段索引策略，则会发生以下情况：

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

使用分面对日志进行分组和浏览

分面对于分析日志非常有用，因为它们允许您以交互方式筛选和分组数据，而无需运行查询。分面是日志中的一个字段（例如`ServiceName`或`StatusCode`），用于跨日志组进行筛选、聚合和分析。您可以在 Lo CloudWatch gs Insights 控制台中查看分面字段列表，以及基于所选时间范围的每个分面值的日志事件计数。当您选择不同的分面和值时，分面值和计数会实时更新，从而使您能够以交互方式浏览日志。

每个方面都显示根据所选时间范围和查询范围自动从日志字段中提取的可用值和计数，并保留 30 天。显示的刻面计数为近似值。您可以使用默认分面（例如数据源名称或数据源类型）来浏览日志，或者在日志中的任何字段上创建自定义分面。数据源名称是生成日志的 AWS 服务或应用程序（例如，Route 53、Amazon VPC 或 CloudTrail），数据源类型是该服务生成的特定日志类型。默认分面由创建 CloudWatch并包括`@aws.region`、`@data_source_name`和`@data_source_type`、和。`@data_format`有关更多信息，请参阅 日志管理。Facets 仅适用于在账户中提取的日志。如果您设置了跨账户可观察性，则监控账户无法根据源账户的日志查看分面。

如果您启用了遥测资源标签，则资源标签字段 (`@aws.tag.*`) 会自动添加为默认分面。这些方面使您能够根据生成日志事件的资源的标签对日志事件进行筛选和分组。

要创建其他方面，请在日志中选择与故障排除相关的字段，然后使用索引策略对其进行配置。对于自定义分面，我们建议在低基数字段（每天具有少于 100 个唯一值的字段，例如 Status 和）上创建它们。ApplicationName每天具有超过 100 个唯一值的刻面被归类为高基数，并且不会显示这些刻面的值。选择一个或多个方面，然后选择在日志中运行查询。

要开始使用 “ CloudWatch 日志见解” 中的各个方面，请执行以下操作：

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、Logs Insights。
3. （可选）使用时间范围选择器选择要分析的时间段。在选定的时间范围内，面板中会显示可用的刻面和值。
4. 选择分面来浏览您的数据，并查看各个方面的价值分布的实时更新。

   不显示唯一值超过 100 个的刻面。要查询特定值，请改为在查询中使用过滤器。

运行基于分面的查询

1. 跨分面选择一个或多个值。
2. 事件计数将根据所选的方面和值进行更新。
3. 选择分面值后，查询范围会更新以反映所选内容。
4. 选择分面值后，选择 run 以执行查询。
5. 每个分面支持的最大唯一值数为 100。例如，如果一个刻面的值超过 100 个，则所有计数都显示为 “-”，表示这些值未知。

保存基于分面的查询

1. 使用一个或多个分面值创建查询。
2. 其余步骤与保存 Logs Insights 查询相同。请参阅保存 CloudWatch 日志见解查询。
3. 您保存的查询可在 “已保存的查询” 部分找到。当您检索已保存的查询时，它将自动包含用于查询的分面和值，从而可以轻松分析日志。

创建账户级分面

1. 要创建分面，您需要先将该字段创建为索引，然后将其配置为分面。在导航窗格中，选择设置、日志、账户级别索引策略。或者，您可以在 “分面” 面板上选择 “管理分面”。
2. 选择 “创建新索引策略”。有关创建索引策略的详细信息，请参阅创建账户级字段索引策略。
3. 要创建分面，请在索引策略创建页面中选中选定字段的 “设置为分面”。

使用 API 进行分面管理

分面管理可以使用字段索引策略完成。有关详细信息，请参阅 `field index`API。

字段索引 API

否。	Name	说明
1	PutIndexPolicy	为特定日志组创建或更新字段索引策略
2	PutAccountPolicy	创建适用于账户中所有日志组或部分日志组的账户级数据保护策略、订阅筛选策略、字段索引策略、转换器策略或指标提取策略
3	DeleteIndexPolicy	删除应用于单个日志组的日志组级别字段索引策略
4	DeleteAccountPolicy	删除 CloudWatch 日志账号策略

在 “日志见解” 中查看周围的 CloudWatch 日志

您可以查看 Logs Insights 查询返回的特定日志记录之前和之后发生的 CloudWatch 日志事件。周围的日志可帮助您了解错误、警告或其他重大事件的背景。

周围日志的工作原理

运行 Lo CloudWatch gs Insights 查询时，结果中的每条日志记录都包含一个 “周围日志” 选项。当您选择此选项时，Lo CloudWatch gs Insights 会显示来自同一日志流的其他日志事件，这些事件发生在选定记录之前和之后。即使这些事件与您的原始查询过滤器不匹配，它们也会出现。

您可以配置要显示的周围日志行数。从所选记录上方和下方的 5、10、20、50 或 100 条日志行中进行选择。

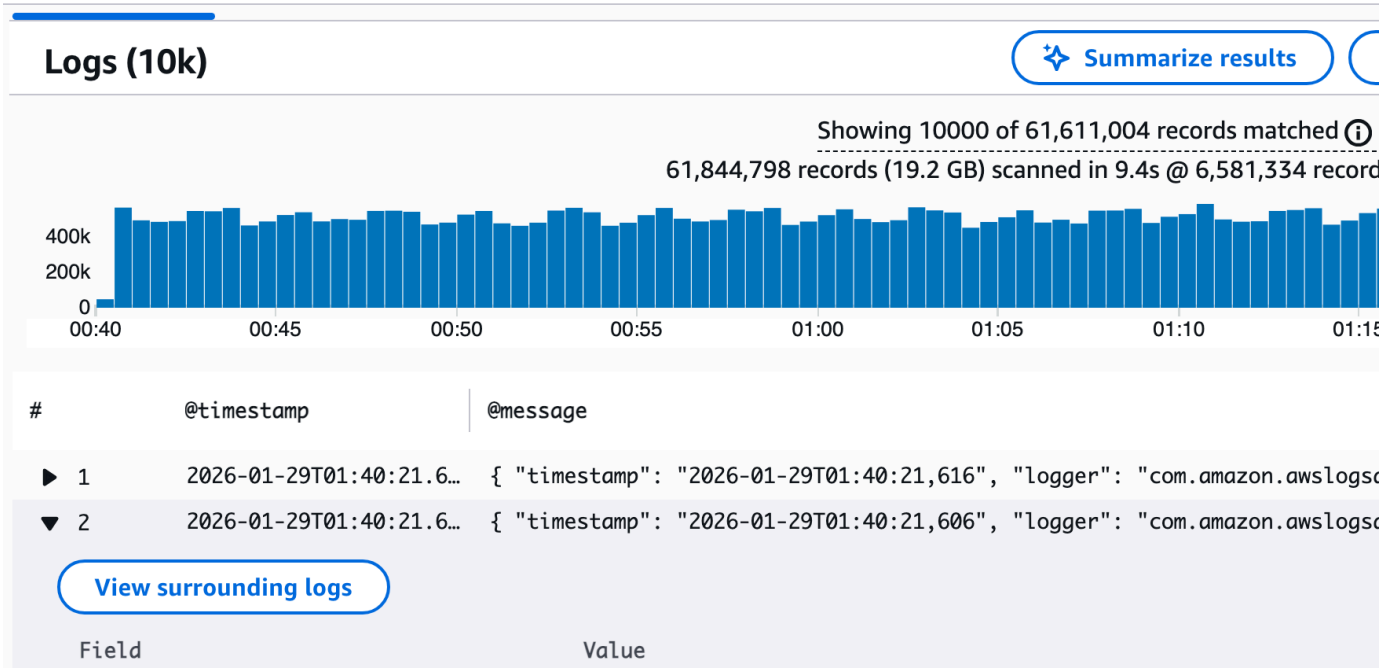
周围的日志在以下情况下很有用：

- 通过查看导致失败的事件来调试错误
- 调查分布式系统中的警告或超时
- 了解应用程序中的事件顺序
- 分析高容量日志环境中的问题

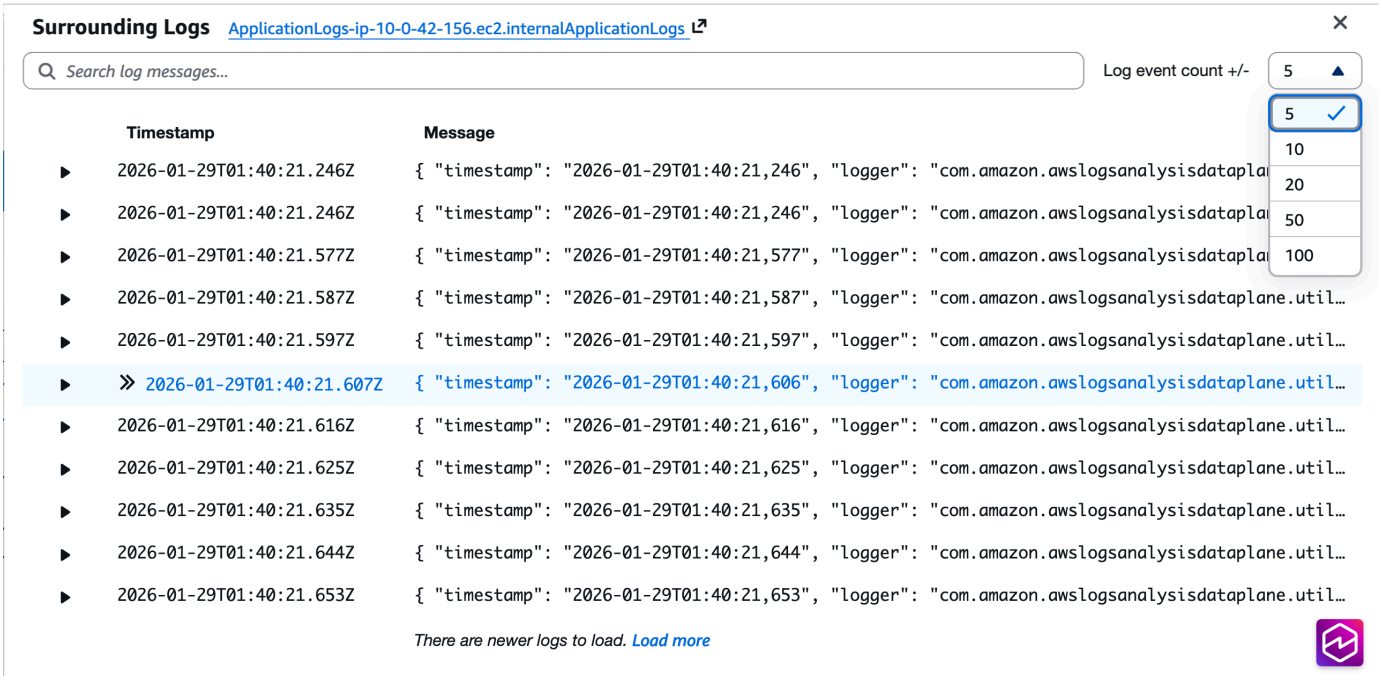
查看周围的日志

查看周围的日志

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，依次选择日志、Logs Insights。
3. 运行您的查询。
4. 在查询结果中，找到要调查的日志记录。
5. 选择 “周围日志”。



6. 在 “事件数量” 下拉列表中，选择要在所选记录上方和下方显示的日志事件数量。你可以选择 5、10、20、50 或 100。

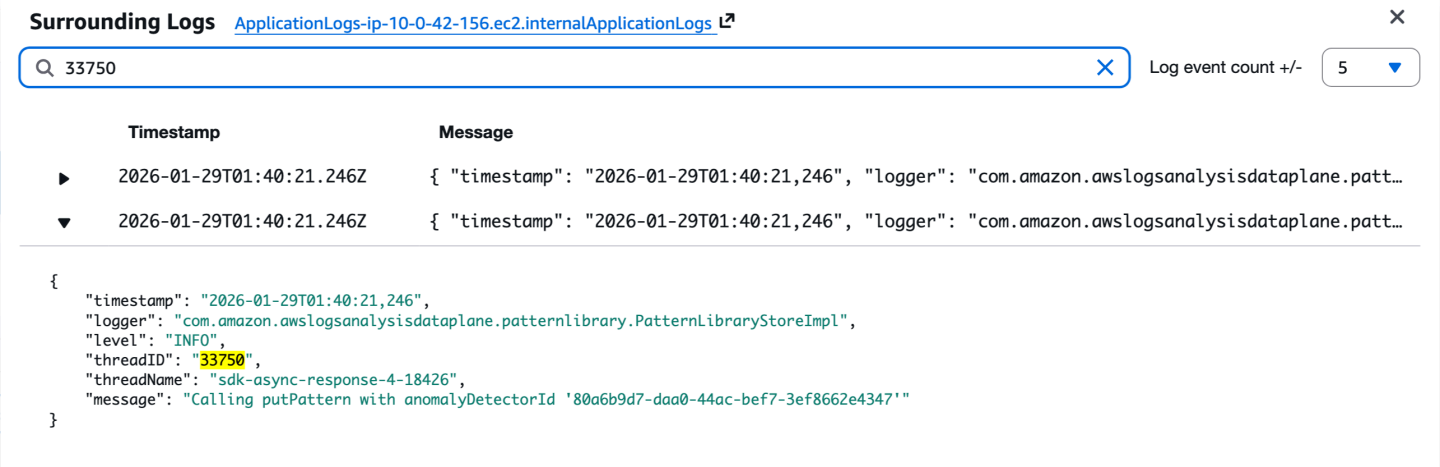


在周围的日志中搜索

打开周围的日志面板后，您可以搜索特定的关键字来查找相关的上下文。

在周围的日志中搜索

1. 在周围的日志面板中，在搜索框中输入您的搜索词。
2. 查看突出显示的匹配日志行。
3. 使用导航控件在比赛之间移动。



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。模式是在日志字段中重复出现的共有的文本结构。查看查询结果时，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，以分析整组匹配日志事件中的模式。

模式对于分析大型日志集很有用，因为大量日志事件通常可压缩成几个模式。

考虑以下三个日志事件示例。

```
2023-01-01 19:00:01 [INFO] Calling DynamoDB to store for resource id 12342342k124-12345
2023-01-01 19:00:02 [INFO] Calling DynamoDB to store for resource id 324892398123-12345
2023-01-01 19:00:03 [INFO] Calling DynamoDB to store for resource id 3ff231242342-12345
```

在上面的示例中，所有三个日志事件都遵循一种模式：

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

模式内的字段称为令牌。模式内变化的字段（如请求 ID 或时间戳）是动态令牌。每个动态令牌都用 <*string*-*number*> 表示。*string*是对令牌所代表的数据类型的描述。*number*显示了与其他动态代币相比，该代币在模式中的显示位置。

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

模式检测还用于 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）。

开始使用模式分析

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

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

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，依次选择日志、Logs Insights。

   在 Logs Insights（日志洞察）页面，查询编辑器包含一个默认查询，它将返回 20 个最近的日志事件。
3. 删除查询框中的 | limit 20 行，使得查询如下所示：

   ```
   fields @timestamp, @message, @logStream, @log
   | sort @timestamp desc
   ```
4. 在选择日志组下拉菜单中，选择要查询的一个或多个日志组。
5. （可选）使用时间间隔选择器选择要查询的时间段。

   您可以选择 5 分钟到 30 分钟的间隔；1 小时、3 小时和 12 小时间隔；或者自定义时间范围。
6. 选择运行查询以启动查询。

   查询运行完毕后，日志选项卡将显示查询返回的日志事件表。表上方会显示一条关于有多少条记录与查询匹配的消息，类似于显示匹配的 10,000 条记录，共 71,101 条。
7. 选择模式选项卡。
8. 该表现在会显示在查询中发现的模式。由于查询不包括 pattern 命令，因此此选项卡仅显示在日志选项卡的表中显示的 10,000 个日志事件中发现的模式。

对于每个模式，将显示下面的信息：

- 模式，每个动态令牌显示为 <*string*-*number*>。*string*是对令牌所代表的数据类型的描述。*number*显示了与其他动态代币相比，该代币在模式中的显示位置。
- 事件计数，这是该模式在查询的日志事件中出现的次数。选择事件计数列标题可按频率对模式进行排序。
- 事件比率，即查询的日志事件中包含此模式的百分比。
- 严重性类型，将是以下项之一：
  - 如果模式包含错误一词，则为 ERROR。
  - 如果模式包含警告一词，但不包含错误，则为 WARN。
  - 如果模式不包含警告或错误，则为 INFO。

  选择严重性信息列标题可按严重性对模式进行排序。

9. 现在更改查询。将查询中的 `| sort @timestamp desc` 行替换为 `| pattern @message`，以便完整查询如下：

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

10. 选择运行查询。

    查询完成后，日志选项卡中没有结果。但是，模式选项卡可能会列出更多模式，具体取决于查询的日志事件总数。

11. 无论您是否在查询中包括 `pattern`，都可以进一步检查查询返回的模式。为此，请在检查列中选择其中一个模式的图标。

    将出现模式检查窗格并显示以下内容：

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

> **Note**
>
> 每个令牌最多可捕获 10 个令牌值。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与您正在检查的模式几乎同时频繁出现的其他模式。例如，如果 ERROR 消息的模式通常伴随另一个标记为 INFO 并带有附加详细信息的日志事件，则此处显示该模式。

有关 pattern 命令的详细信息

本节包含有关 pattern 命令及其用途的更多详细信息。

- 在上一个教程中，我们在添加 pattern 时删除了 sort 命令，因为如果查询在 sort 命令之后包含 pattern 命令，则查询无效。在 sort 排序之前包含 pattern 是有效的。

  有关 pattern 语法的更多详细信息，请参阅 模式。
- 当在查询中使用 pattern 时，@message 必须是 pattern 命令中选择的字段之一。
- 可以在 pattern 命令之前包含 filter 命令，以便仅将筛选后的日志事件集用作模式分析的输入。
- 要查看特定字段（例如从 parse 命令派生的字段）的模式结果，请使用 pattern @fieldname。
- 具有非日志输出的查询（例如使用 stats 命令的查询）不会返回模式结果。

保存并重新运行 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。查询将以文件夹结构保存，因此您可以组织它们。每个账户每个区域可以保存多达 1000 个查询。

查询保存在某个 Region-specific 级别，而不是用户特定的级别。如果您创建并保存查询，则有权访问同一区域 CloudWatch 日志的其他用户可以在该区域中看到所有已保存的查询及其文件夹结构。

要保存查询，您必须登录到具有 logs:PutQueryDefinition 权限的角色。要查看已保存查询的列表，您必须登录到具有 logs:DescribeQueryDefinitions 权限的角色。

> **Note**
>
> 您可以使用参数创建和保存查询，这些参数是带有命名占位符的可重复使用的模板。与其保存具有不同值的同一个查询的多个变体，不如创建一个模板并在运行时提供不同的参数值。目

前，仅使用 Logs Insights 查询语言的查询支持此功能。有关更多信息，请参阅使用带参数的已保存查询。

Console

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。
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. （可选）要在查询中使用参数，请执行以下额外步骤：
   a. 在查询中添加参数。使用{{parameter}}语法（参数名称前后的双大括号）将静态值替换为占位符。

      示例：具有静态值的原始查询：

      ```
      fields @timestamp, @message
      | filter level = "Error"
      | filter applicationName = "OrderService"
      ```

      使用参数更新了查询：

      ```
      fields @timestamp, @message
      | filter level = {{logLevel}}
      | filter applicationName = {{applicationName}}
      ```

   b. 定义查询中使用的参数。对于每个占位符参数，请指定：
      - 名称：必须与占位符名称完全匹配（例如logLevel、applicationName）。

- 默认值（可选）：未提供参数值时使用的值。
- 描述（可选）：解释参数的用途。

c. 通过使用带$前缀的查询名称并将参数名称作为键值对传递，可以运行带参数的查询。有关详细信息，请参阅运行保存的查询。

9. 选择保存。

AWS CLI

要保存查询，请使用put-query-definition：

```
aws logs put-query-definition \
  --name "ErrorsByLevel" \
  --query-string "fields @timestamp, @message | filter level = \"ERROR\"" \
  --log-group-names "/aws/lambda/my-function" \
  --region us-east-1
```

（可选）要保存带参数的查询，请添加--parameters选项并在查询字符串中使用{{parameterName}}占位符：

```
aws logs put-query-definition \
  --name "ErrorsByLevel" \
  --query-string "fields @timestamp, @message | filter level = {{logLevel}} | filter
 applicationName = {{applicationName}}" \
  --parameters '[{"name":"logLevel","defaultValue":"ERROR","description":"Log level
 to filter"},
{"name":"applicationName","defaultValue":"OrderService","description":"Application
 name to filter"}]' \
  --log-group-names "/aws/lambda/my-function" \
  --region us-east-1
```

要将查询保存在文件夹中，请在查询名称前加上文件夹路径：

```
aws logs put-query-definition \
  --name "my-folder/ErrorsByLevel" \
  --query-string "fields @timestamp, @message | filter level = {{logLevel}}" \
  --parameters '[{"name":"logLevel","defaultValue":"ERROR","description":"Log level
 to filter"}]' \
  --log-group-names "/aws/lambda/my-function" \
  --region us-east-1
```

API

要保存查询，请致电 PutQueryDefinition：

```
{
  "name": "ErrorsByLevel",
  "queryString": "fields @timestamp, @message | filter level = \"ERROR\"",
  "logGroupNames": ["/aws/lambda/my-function"]
}
```

（可选）要保存带参数的查询，请在查询字符串中包含该parameters字段并使用{{parameterName}}占位符：

```
{
  "name": "ErrorsByLevel",
  "queryString": "fields @timestamp, @message | filter level = {{logLevel}} | filter
 applicationName = {{applicationName}}",
  "logGroupNames": ["/aws/lambda/my-function"],
  "parameters": [
    {
      "name": "logLevel",
      "defaultValue": "ERROR",
      "description": "Log level to filter"
    },
    {
      "name": "applicationName",
      "defaultValue": "OrderService",
      "description": "Application name to filter"
    }
  ]
}
```

> **Tip**
>
> 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。

Console

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“已保存的查询”列表中选择您的查询。查询文本出现在查询编辑器中。
5. （可选）要使用带参数的查询，请执行以下操作：
   a. 在“已保存的查询”侧面板中，选择查询名称旁边的 + 图标。
   b. 带有参数的查询出现在查询编辑器中。例如，如果您选择旁边的 + 图标`ErrorsByLevel`，则查询编辑器中将填充以下内容：`$ErrorsByLevel(level=, applicationName=)`
   c. 提供参数（级别、应用程序名称）的值并运行查询。例如：`$ErrorsByLevel(level= "ERROR", applicationName= "OrderService")`
6. 选择运行。

AWS CLI

使用参数运行已保存的查询

`start-query`使用以下`$QueryName()`语法：

```
aws logs start-query \
  --log-group-names "/aws/lambda/my-function" \
  --start-time 1707566400 --end-time 1707570000 \
  --query-string '$ErrorsByLevel(level= "ERROR", applicationName= "OrderService")' \
  --region us-east-1
```

API

使用参数运行已保存的查询

StartQuery使用`queryString`字段中的`$QueryName()`语法调用：

```
{
```

```
  "logGroupNames": ["/aws/lambda/my-function"],
  "startTime": 1707566400,
  "endTime": 1707570000,
  "queryString": "$ErrorsByLevel(level=\"ERROR\", applicationName= \"OrderService
\")"
}
```

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）。

使用带参数的已保存查询

带有参数的已保存查询是带有命名占位符的可重复使用的查询模板。您可以保存模板并在运行查询时提供不同的参数值，而不必维护几乎相同的查询的多个副本。只有 L CloudWatch ogs Insights 查询语言支持参数。

工作原理

保存查询时，占位符会标识您在执行查询时可以提供的值。占位符使用{{parameterName}}语法。以下是一个名为的已保存查询的示例，该查询名ErrorsByLevel为两个参数logLevel和applicationName。

```
fields @timestamp, @message
| filter level = {{logLevel}}
| filter applicationName = {{applicationName}}
```

要运行已保存的查询，您可以使用前缀为前缀的查询名称来调用该查询，$并传递参数值。CloudWatch Logs Insights 查询引擎会替换每个占位符。如果参数包含默认值，则如果未提供其他值，则使用这些值。

```
# Run query by using query name and passing parameter values explicitly
$ErrorsByLevel(logLevel = "WARN", applicationName = "OrderService")

# Run query without specifying parameter values - default values are used in this case.
$ErrorsByLevel()
```

包含空格或特殊字符的已保存查询名称需要用反引号括起来：

```
$`Errors By Level`(logLevel = "WARN")
```

带参数的已保存查询示例

添加结果限制作为参数

查询名称：ErrorsByLevel带参数logLevel（默认："ERROR"）、applicationName（默认："OrderService"）和maxResults（默认：50）

```
fields @timestamp, @message, @logStream
| filter level = {{logLevel}}
| filter applicationName = {{applicationName}}
| sort @timestamp desc
```

```
| limit {{maxResults}}
```

```
# Run the query using the query name and passing parameter values
$ErrorsByLevel(logLevel = "WARN", applicationName = "OrderService", maxResults = 100)
```

使用多个已保存的带参数的查询

下面的示例使用ErrorsByLevel了第二个保存的查询RecentN，该查询定义为sort @timestamp desc | limit {{count}}（带参数count，默认20）。 CloudWatch Logs Insights 查询引擎会在运行每个查询之前对其进行扩展。

```
# Using multiple queries with parameters in sequence
$ErrorsByLevel(logLevel = "WARN", applicationName = "OrderService")
| $RecentN(count = 10)

# Each of the queries is expanded, resulting in the following query when it is run.
fields @timestamp, @message
| filter level = "WARN"
| filter applicationName = "OrderService"
| sort @timestamp desc
| limit 10
```

配额和错误处理

> **Note**
>
> 每个保存的查询最多可以有 20 个参数。

扩展后的查询字符串不能超过 10,000 个字符。参数名称必须以字母或下划线开头。已保存的查询不能引用其他已保存的查询（不支持嵌套调用）。

常见错误

错误	原因
只有 CWLI 查询语言支持参数	只有 L CloudWatch ogs Insights 查询语言支持参数。
在 QueryString 中找不到必需的参数	中的参数名称在查询字符串{{placeholder}} 中--parameters 没有匹配项。

错误	原因
参数计数超过最大值 20	保存的查询目前仅支持 20 个参数。
重复的参数名	查询定义中有重复的参数`parameters` 。

> **Note**
>
> 要使用参数创建或更新已保存的查询，您需要该`logs:PutQueryDefinition`权限。要运行一个，你需要`logs:StartQuery`和`logs:DescribeQueryDefinitions`。

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。这些查询计入您并发 100 个 L CloudWatch ogs Insights 查询的上限。

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 100 个 CloudWatch Logs Insights 查询，包括添加到仪表板的查询。此外，您可以为 OpenSearch 服务 PPL 或 OpenSearch 服务 SQL 运行 15 个并发查询。

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，使用以下 `policy.json`get-key-policy 命令将您的 KMS 密钥默认策略保存为 ：

```
aws kms get-key-policy --key-id key-id --policy-name default --output text > ./
policy.json
```

在文本编辑器中打开 `policy.json` 文件，然后从以下语句之一中添加粗体的部分。使用逗号将现有语句与新语句分隔。这些语句使用`Condition`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。

JSON

```
{
    "Version":"2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
```

```
            "Principal": {
                "Service": "logs.region.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt*",
                "kms:Decrypt*",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:Describe*"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                "aws:SourceArn": "arn:aws:logs:us-east-1:111122223333:query-
result:*"
                },
                "StringEquals": {
                "aws:SourceAccount": "111122223333"
                }
            }
        }
    ]
}
```

最后，使用以下 put-key-policy 命令添加更新后的策略：

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

步骤 3：将 KMS 密钥与查询结果关联

将 KMS 密钥与账户中的查询结果相关联

按如下方式使用 disassociate-kms-key 命令：

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

步骤 4：取消密钥与账户中查询结果的关联

要取消 KMS 密钥与查询结果的关联，请使用以下 disassociate-kms-key 命令：

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

根据 Lo CloudWatch gs Insights 查询结果生成自然语言摘要

分析日志数据对于了解应用程序的行为至关重要，但是解释大量日志条目可能很耗时。 CloudWatch Logs Insights 现在提供自然语言摘要功能，可将复杂的查询结果转换为清晰、简洁的摘要。此功能可帮助您快速识别问题，并从日志数据中获得切实可行的见解。

工作原理

CloudWatch Logs Insights 可以使用 Amazon Bedrock 从您的查询结果中生成人类可读的摘要。该功能支持所有 CloudWatch Logs Insights 查询语言，并从您的日志数据中提供清晰、可操作的见解。

区域可用性和数据处理

> **Important**
>
> 使用此功能时，您的查询结果可能会在不同的 AWS 区域进行处理。例如，如果您在美国东部（弗吉尼亚州北部）运行查询，则汇总信息可能会在美国西部（俄勒冈州）进行处理。

下表列出了可 AWS 区域 使用查询结果功能的不同地区可能的处理方式：

支持的 CloudWatch 日志地理位置	可能的处理区域
美国（US）	US East (N. Virginia) Region 美国东部（俄亥俄）区域 US West (Oregon) Region
欧洲	欧洲地区（法兰克福）区域 欧洲地区（爱尔兰）区域 欧洲地区（巴黎）区域 欧洲地区（斯德哥尔摩）区域

支持的 CloudWatch 日志地理位置	可能的处理区域
	欧洲地区（伦敦）区域
亚太地区	US East (N. Virginia) Region 美国东部（俄亥俄）区域 US West (Oregon) Region
南美洲	US East (N. Virginia) Region 美国东部（俄亥俄）区域 US West (Oregon) Region

开始使用

生成自然语言摘要

1. 运行 “ CloudWatch 日志见解” 查询。
2. 查询完成后，选择汇总结果。

Permissions

您必须拥有以下其中一项：

- `CloudWatchLogsFullAccess` 许可
- `CloudWatchLogsReadOnlyAccess` 许可
- 自定义 IAM 策略，包括 `cloudwatch:GenerateQueryResultsSummary`、`logs:GetQueryResults`、`logs:DescribeQuer` 和 `logs:FilterLogEvents` 操作

数据隐私

您的查询结果会得到安全处理，不会用于训练或改进 L CloudWatch ogs Insights 或 Amazon Bedrock。如果您选择使用反馈按钮对查询结果摘要提供反馈，则您的反馈表明您对 L CloudWatch ogs Insights 中提供的功能的满意程度。

使用计划查询自动进行日志分析

计划查询使您能够定期自动执行日志查询。您可以将计划查询配置为自动运行并将结果传送到 Amazon S3 存储桶、Amazon EventBridge 事件总线或查询表等目的地，而不是手动运行查询来分析日志数据。这种自动化非常适合生成定期报告、监控趋势、使用最新的日志数据保持查询表的最新状态，或者根据日志分析结果触发下游流程。

计划查询支持所有三种可用于日志查询的查询语言：

- 日志见解查询语言（Logs Insights QL）
- OpenSearch 服务管道处理语言 (PPL)
- OpenSearch 服务结构化查询语言 (SQL)

内容

- 了解计划查询的概念
- 时间表表达式参考
- 最佳实践
- 计划查询入门
- 为计划查询配置 S3 目标
- 为计划查询配置查找表目的地
- 对计划查询进行故障排除

了解计划查询的概念

在创建计划查询之前，请了解这些关键概念，这些概念会影响查询的运行方式和结果的交付地点。

IAM 角色分离

计划查询需要两个独立的 IAM 角色：一个用于执行查询，另一个用于将结果传送到 Amazon S3 存储桶、Amazon EventBridge 事件总线或查询表等目的地。了解这种分离的原因有助于您正确配置权限并使用其提供的安全和运营优势。

双角色架构在数据访问和数据交付之间划分了责任。查询执行角色访问您的日志数据并运行查询，而目标交付角色将结果写入您选择的目的地。这种分离遵循最小权限的原则——每个角色仅拥有其特定功能所需的权限。

查询执行角色

允许 CloudWatch 日志代表您运行 CloudWatch 日志见解查询。此角色需要访问您的日志组和执行查询的权限，但不需要访问目标资源。所需权限：

- logs:StartQuery
- logs:StopQuery
- logs:GetQueryResults
- logs:DescribeLogGroups
- logs:Unmask是否需要取消屏蔽数据

对于 KMS-encrypted 日志组：kms:Decrypt以及用于加密日志组的 KMS 密钥的kms:DescribeKey权限。还需要添加这些权限。

信任关系要求：查询执行角色必须包含允许 CloudWatch 日志服务 (logs.amazonaws.com) 代入该角色的信任策略。如果没有这种信任关系，计划查询将因权限错误而失败。

查询执行角色的信任策略示例：

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

查询执行角色的权限策略示例：

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

```
                "logs:StopQuery",
                "logs:GetQueryResults",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

目的地配送角色

允许 CloudWatch 日志将查询结果传送到您选择的目的地。遵循最小权限原则，此角色只需要特定目标服务的权限。所需的权限因目的地类型而异。

信任关系要求：目标交付角色还必须包含允许 CloudWatch 日志服务 (`logs.amazonaws.com`) 代入该角色的信任策略。

S3 目标交付角色的权限策略示例：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::your-scheduled-query-results-bucket/*"
        }
    ]
}
```

查找表目标交付角色的权限策略示例：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLookupTable",
                "logs:UpdateLookupTable",
                "logs:GetQueryResults"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

这种分离为您的运营带来了实际好处。从安全角度来看，如果您需要更改结果的交付地点，则只需修改目标交付角色而不更改查询执行权限。对于合规性和审计，您可以清楚地跟踪哪个角色访问敏感日志数据以及哪个角色写入外部系统。这样可以更轻松地证明您的日志分析基础架构遵循安全最佳实践。

Cross-region 以及跨账户的使用

计划查询是在特定区域创建的，并在该区域运行。但是，您可以查询日志组并跨区域和跨账户提供结果。您需要将一个或多个 AWS 账户设置为监控账户，并将它们与多个来源账户关联起来。监控账户是一个中央账户，可以查看源 AWS 账户生成的可观测性数据并与之交互。源账户是一个个人 AWS 账户，它为其中的资源生成可观测性数据。源账户与监控账户共享其可观测性数据。因此，您可以使用所有关联账户的日志组设置来自监控账户的预定查询。

查询跨区域日志组

您的预定查询可以访问任何区域的日志组。使用其完整 ARN 格式指定日志组：`arn:aws:logs:region:account-id:log-group:log-group-name`。所有目标区域中日志组的查询执行角色需要`logs:StartQuery`和`logs:GetQueryResults`权限。

> **Important**
>
> 在查询日志组或跨区域交付结果时，日志数据会跨越区域边界。请考虑以下事项：
>
> - 数据驻留要求 -确保跨区域数据传输符合贵组织的数据治理政策和监管要求
> - 数据传输成本 - Cross-region 数据传输会产生额外费用
> - 网络延迟 -访问远程区域日志组的查询可能会遇到更高的延迟
>
> 为了获得最佳性能和成本效益，请在与主日志组相同的区域中创建计划查询。

替代方法：使用CloudWatch 日志集中化将来自多个账户和地区的日志数据复制到中央监控账户。这使您可以在单个区域中创建计划查询以访问所有集中日志，从而避免跨区域查询并简化 IAM 权限管理。

调度表达式和时区处理

您定义的时间表决定了查询的运行时间和执行频率。选择正确的计划表达式会影响您何时收到结果以及查询的数据量。了解表达式类型有助于您在简单性和精确性之间做出选择。

Cron 表达式可以精确控制时间，允许您指定确切的时间、一周中的几天或一月中的某天。当您需要在特定工作时间运行查询或与运营计划保持一致时，请使用 cron 表达式。在控制台中，您还可以使用简易日历选项安排查询。

Cron 表达式

在特定时间运行查询。格式：`cron(minute hour day-of-month month day-of-week year)`。示例：

- `cron(0 9 * * ? *)`-世界标准时间每天上午 9:00
- `cron(0 18 ? * MON-FRI *)`-世界标准时间工作日下午 6:00
- `cron(0 0 1 * ? *)`-世界标准时间每个月的第一天午夜
- `cron(0 12 ? * SUN *)`-世界标准时间每周日中午
- `cron(30 8 1 1 ? *)`-世界标准时间 1 月 1 日上午 8:30

所有计划查询均以 UTC 运行，无论您的本地时区如何，也无论您的 AWS 资源位于何处。当您为工作时间或时间敏感型分析安排查询时，这一点尤其重要。例如，如果您的企业在美国东部时间运营，并且您想在美国东部时间上午 9 点提交每日报告，则需要考虑 UTC 偏移量（夏令时为 14:00 UTC，否则为 13:00 UTC）。在规划日程表达式时要牢记 UTC，确保查询在预定时间运行。

选择查询语言

计划查询支持三种不同的查询语言，您的选择会影响您编写查询的方式以及团队维护查询的难易程度。正确的语言取决于您的分析要求和团队的现有技能。

如果您主要筛选和聚合日志数据， CloudWatch Logs Insights 查询语言可提供最直接的语法。对于需要通过多个步骤重塑或丰富数据的复杂数据转换，PPL 的流水线方法使逻辑更易于理解。当您需要执行类似于数据库操作的联接或复杂聚合时，SQL 会提供熟悉的语法，数据库经验丰富的团队可以快速采用这些语法。

CloudWatch 日志见解查询语言 (CWLI)

Purpose-built 使用直观的语法进行日志分析。最适合：

- Text-based 日志分析和过滤
- Time-series 汇总和统计
- 刚接触日志分析的团队

OpenSearch 服务管道处理语言 (PPL)

Pipeline-based 具有强大数据转换功能的查询语言。最适合：

- 复杂的数据转换和充实
- Multi-step 数据处理工作流程
- 熟悉基于管道的处理的团队

OpenSearch 服务结构化查询语言 (SQL)

熟悉的数据库式查询的标准 SQL 语法。最适合：

- 复杂的连接和聚合
- 商业智能和报告
- 具有丰富的 SQL 经验的团队

目的地选择和用例

将查询结果发送到何处决定了您可以对它们做什么。无论您是在构建长期分析、触发自动响应，还是两者兼而有之，这种选择都会影响您的整个下游工作流程。了解每种目的地类型的优势有助于您为用例设计正确的架构。

Amazon S3 目的地针对存储和批处理进行了优化。当您需要将查询结果保存数月或数年、分析一段时间内的趋势或向分析平台提供数据时，Amazon S3 可提供具有无限保留期的经济实惠的存储。EventBridge 目的地已针对实时自动化进行了优化。当查询结果应触发即时操作（例如发送警报、启动工作流程或更新系统）时，将结果作为事件EventBridge 提供，您的应用程序可以立即做出响应。默认情况下，所有查询完成事件都会自动作为事件发送到默认事件总线，从而可以与下游处理系统、Lambda 函数或其他事件驱动架构集成。只有成功执行查询后，结果才会发布到目的地。查询表目的地经过优化，可使参考数据保持最新状态。查找表目标会在每次计划执行时使用查询结果自动填充或刷新指定的查找表，因此其他查询可以使用该命令引用最新数据。`lookup`

Amazon S3 目标

将查询结果存储为 JSON 文件，以便长期保留和批处理。Amazon S3 目的地最适合以下场景：

- 历史分析和数据存档

- 与数据湖和分析平台集成
- 合规和审计要求
- Cost-effective 存储大型结果集

EventBridge 目的地

将查询结果作为事件发送，以进行实时处理和自动化。在事件`queryId`中使用来检索查询结果，查询运行后 30 天内仍可用。 EventBridge目的地最适合以下情况：

- 触发对查询结果的自动响应
- 与无服务器工作流程和 Lambda 函数集成
- Real-time 警报和通知系统
- Event-driven 架构和微服务

查找表目的地

在每次计划执行时自动创建或刷新包含查询结果的查找表。每次刷新都是表格内容的完全替换。查找表目的地最适合以下场景：

- 在日志查询中保持`lookup`命令的参考数据是最新的
- 维护源自日志数据的许可名单、拒绝名单或实体清单
- 使用最近的活动摘要（例如活跃用户或资源列表）丰富查询

查询结果的格式和结构

计划查询以 JSON 格式提供结果，但每种目标类型接收的负载不同。对于查找表目的地，查询结果成为查找表的内容，每次运行都会替换该内容。有关更多信息，请参阅 为计划查询配置查找表目的地。

Amazon S3 目标接收查询的结果行。每个对象都包含一个 JSON 数组，结果集中每行都有一个条目，每个条目将查询的输出字段名称映射到其值。该对象不包含查询元数据或查询统计信息，即使查询请求它也会省略该`@ptr`字段，因为该字段只能在控制台中使用。

EventBridge 目标接收查询元数据，包括查询统计信息，但不接收结果行。要检索已完成查询的行，请GetQueryResults使用事件`queryId`中的值进行调用。

以下示例显示了计划查询完成 EventBridge 时 CloudWatch 日志发布到的事件。

```
{
    "version": "0",
```

```
    "id": "be72061b-eca2-e068-a7e1-83e01d6fe807",
    "detail-type": "Scheduled Query Completed",
    "source": "aws.logs",
    "account": "123456789012",
    "time": "2025-11-18T11:31:48Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:logs:us-east-1:123456789012:scheduled-query:477b4380-b098-474e-9c5e-
e10a8cc2e6e7"
    ],
    "detail": {
        "queryId": "2038fd57-ab4f-4018-bb2f-61d363f4a004",
        "queryString": "fields @timestamp, @message, @logStream\n| filter @message
 like /ERROR/\n| sort @timestamp desc\n| limit 10000",
        "logGroupIdentifiers": [
            "/aws/lambda/my-function"
        ],
        "status": "Complete",
        "startTime": 1763465460,
        "statistics": {
            "recordsMatched": 1842,
            "recordsScanned": 48325,
            "estimatedRecordsSkipped": 0,
            "bytesScanned": 12081250,
            "estimatedBytesSkipped": 0,
            "logGroupsScanned": 1,
            "resultCount": 1842
        }
    }
}
```


此查询不聚合，返回的行数少于 10,000 行，因此 1,842 个匹配的日志事件中的每一个`resultCount`都成为一个输出行，`recordsMatched`且相等。`limit`聚合查询或结果集被截断的查询生成的值`limit`小`resultCount`于。`recordsMatched`有关更多信息，请参阅 了解已扫描的记录、匹配的记录和结果数。

关键要素包括：

- `statistics`-描述查询读取的日志数据量以及结果集大小的计数器。有关每个字段的描述，请参阅下表。
- `startTime`-开始执行查询的时间（Unix 时间戳）
- `queryString`-执行的实际查询

- queryId-查询的查询 ID，使用它可以检索结果
- logGroupIdentifiers-查询的日志组列表
- status-查询执行状态（完成、失败等）

下表描述了statistics对象中的每个字段。有关这些字段的 API 定义，请参阅QueryStatistics。

字段	说明
recordsScanned	查询期间扫描的日志事件总数。
recordsMatched	与查询字符串匹配的日志事件的数量。此值计算日志事件，而不是输出行。对于结果集中的行数，使用resultCount 。
resultCount	查询结果集中的行数。该值仅计算在查询中所有操作中幸存下来的行，因此它可能小于recordsMatched 。它涵盖了GetQueryResults 返回的所有结果页面。有关更多信息，请参阅 了解已扫描的记录、匹配的记录和结果数。
estimatedRecordsSkipped	处理此查询时跳过的日志事件数量的估计值，因为该查询包含索引字段。跳过这些条目可以降低查询成本并缩短查询性能时间。有关更多信息，请参阅 创建字段索引以提高查询性能并减少扫描量。
bytesScanned	查询期间扫描的日志事件中的总字节数。
estimatedBytesSkipped	由于查询包含索引字段，因此在处理此查询时跳过的日志事件中的字节数的估计值。
logGroupsScanned	此查询扫描的日志组的数量。

了解已扫描的记录、匹配的记录和结果数

其中三个查询统计数据计算不同的内容，直接比较可能会产生误导。每一个都衡量查询处理的不同阶段：

- recordsScanned-查询从您的日志组读取的日志事件的数量。这是查询的输入。
- recordsMatched-与查询字符串匹配的日志事件的数量。此值计算日志事件。
- resultCount-查询生成的结果集中的行数。此值计算输出行数。

每个阶段都会缩小数据范围。诸如之类的命令stats将许多日志事件合并到一个输出行中，因此resultCount可以小得多recordsMatched。大recordsMatched加小resultCount并不意味着结果集中缺少行。

示例-聚合

以下查询对一小时内一个日志组的每个日志流中的错误消息进行计数。

```
filter @message like /ERROR/
| stats count(*) as errorCount by @logStream
```

如果查询读取 1,500,000 个日志事件，其中 24,318 个事件包含ERROR，并且匹配的日志事件来自 12 个日志流，则查询将以以下统计信息完成。

```
"statistics": {
    "recordsMatched": 24318,
    "recordsScanned": 1500000,
    "estimatedRecordsSkipped": 0,
    "bytesScanned": 450000000,
    "estimatedBytesSkipped": 0,
    "logGroupsScanned": 1,
    "resultCount": 12
}
```

stats为每个日志流生成一行，所以resultCount是 12，而recordsMatched是 24,318。的 12 个值errorCount加起来等于 24,318。

示例 — Post-aggregation 过滤器

以下查询仅保留产生超过 1,000 个错误的日志流。

```
filter @message like /ERROR/
| stats count(*) as errorCount by @logStream
| filter errorCount > 1000
```

最后一个filter命令在分组后运行，因此它会从结果集中删除行，而不是从扫描中删除日志事件。如果 12 个日志流中的 3 个errorCount大于 1,000，则resultCount是 3 而不是 12。recordsScanned并且bytesScanned不要更改，因为无论哪种方式，查询都会读取相同的日志数据。

示例 — 极限

limit命令resultCount无需任何聚合即可减少。以下查询返回 100 条最新错误消息。

```
filter @message like /ERROR/
| sort @timestamp desc
| limit 100
```

如果查询扫描相同的 1,500,000 个日志事件并匹配相同的 24,318 个，则resultCount为 100，因为结果集的limit上限为 100 行。在控制台中，你会看到这种关系显示了 24,318 条匹配的记录中的 100 条。

每个统计数据都回答了一个不同的问题。

这个查询返回了多少行？

使用 resultCount。值为 0 表示该查询未生成任何行。请勿recordsMatched用于此目的，因为它计算日志事件而不是行数。

这个查询读取了多少日志数据？

使用 recordsScanned 和 bytesScanned。扫描量决定查询的成本和运行时间。要缩短时间范围，减少查询日志组的次数，或者创建字段索引。有关更多信息，请参阅 创建字段索引以提高查询性能并减少扫描量。

有多少日志事件与该查询相匹配？

使用 recordsMatched。

> **Note**
>
> CloudWatch 当对象没有可用值时，日志会省略该statistics对象的统计数据，而不是将统计数据报告为 0。
> 如果您的事件使用者在事件resultCount中找不到，则将该值视为未知值而不是 0。编写事件使用者，让他们容忍不存在的统计数据，忽略他们无法识别的统计数据。

时间表表达式参考

使用这些参考表为您的计划查询构造计划表达式。所有时间均采用 UTC。

Cron 表达式语法

格式：`cron(minute hour day-of-month month day-of-week year)`

使用场景	Cron 表达式	说明	何时使用
每日日程安排	`cron(0 9 * * ? *)`	世界标准时间每天上午 9:00	每日报告
	`cron(0 */6 * * ? *)`	每 6 小时（UTC 时间 00:00、06:00、12:00、18:00）	频繁监控
	`cron(30 2 * * ? *)`	世界标准时间每天凌晨 2:30	Off-peak 分析
营业时间	`cron(0 9-17 ? * MON-FRI *)`	UTC 周一至周五上午 9 点至下午 5 点每小时一班	业务监控
	`cron(0 18 ? * MON-FRI *)`	世界标准时间工作日下午 6:00	工作日结束
	`cron(0 8,12,17 ? * MON-FRI *)`	UTC 工作日上午 8 点、中午和下午 5 点	关键营业时间
每周日程安排	`cron(0 12 ? * SUN *)`	世界标准时间每周日中午	每周摘要
	`cron(0 9 ? * MON *)`	世界标准时间每周一上午 9:00	本周开始报告
	`cron(0 23 ? * FRI *)`	世界标准时间每周五晚上 11:00	周末大扫除
每月日程表	`cron(0 0 1 * ? *)`	世界标准时间每个月的第一天午夜	月度报告
	`cron(0 9 L * ? *)`	世界标准时间每个月的最后一天上午 9:00	月末处理
	`cron(0 10 1 1,4,7,10 ? *)`	每个季度的第一天世界标准时间上午 10:00	季度分析

使用场景	Cron 表达式	说明	何时使用
高频	`cron(*/15 * * * ? *)`	每 15 分钟一次	Real-time 监控
	`cron(0,30 * * * ? *)`	每 30 分钟（在 00:00 和 30 分）	经常检查
	`cron(0 */2 * * ? *)`	每 2 小时	定期间隔
特殊案例	`cron(30 8 1 1 ? *)`	世界标准时间 1 月 1 日上午 8:30	年度报告
	`cron(0 6 * * SAT,SUN *)`	世界标准时间周末上午 6:00	周末处理
	`cron(0 0 ? * MON#1 *)`	每个月的第一个星期一午夜 UTC	月度计划

Cron 表达式字段参考

字段	值	通配符	示例
分钟（第 1 分钟）	0-59	`* , - /`	`0`（每小时），`*/15`（每 15 分钟），`0,30`（每小时两次）
小时（第 2 次）	0-23	`* , - /`	`9`（上午9点），`*/2`（每2小时一次），`9-17`（营业时间）
Day-of-month（第 3 个）	1-31，L，W	`* , - / ?`	`1`（第 1 天）、`L`（最后一天）、`?`（使用工作日时）
月（第 4 个）	1-12 或 JAN-DEC	`* , - /`	`1`（一月），`JAN`，`1,4,7,10`（每季度）

字段	值	通配符	示例
Day-of-week（第 5 名）	1-7 或 SUN-SAT	* , - / ? # L	MON-FRI（平日），SUN，MON#1（第一个星期一）
年份（第 6 年）	1970-2199	* , - /	*（每年），2024（特定年份），2024-2026 （范围）

通配符和特殊表达式

*****（星号）

匹配该字段中的所有值。示例：*在小时字段中表示每小时。

?（问号）

没有具体值。如果指定了其他选项，则在月中某一天或一周中的某一天使用。示例：指定星期几时，以月?中的某一天为单位使用MON-FRI。

-（短划线）

值的范围。示例：MON-FRI（周一至周五），9-17（上午 9 点至下午 5 点）。

,（逗号）

多个特定值。示例：MON,WED,FRI（星期一、星期三、星期五），8,12,17（上午 8 点、中午、下午 5 点）。

/（斜线）

步长值或增量。示例：0/15以分钟为单位表示从第 0 分钟（0、15、30、45）开始每 15 分钟一次。*/2以小时为单位表示每 2 小时一次。

L（最后）

一个月的最后一天或工作日的最后一次出现。示例：L以月日表示本月的最后一天。FRIL表示本月的最后一个星期五。

W（工作日）

最近的工作日。示例：15W表示距离当月 15 日最近的工作日。

#（第 n 次出现）

本月工作日第 n 次出现。示例：MON#1表示每月的第一个星期一，FRI#2表示每月的第二个星期五。

常见模式和最佳实践

- 对于业务应用程序：使用MON-FRI工作时间（例如9-17）以避免在周末或非工作时间运行查询。
- 对于高频监控：使用增量，例如*/15（每 15 分钟），但要注意查询并发限制。
- 为了提高资源效率：使用 UTC 等清晨时间在非高峰时段安排资源密集型查询。2-6
- 对于月度报告：L用于每月的最后一天或特定日期（例如1第一天），以确保时间一致。

最佳实践

遵循以下最佳实践来确保可靠和高效的计划查询操作：

查询优化

- 在调度之前手动测试查询以验证性能和结果
- 在查询的早期使用筛选索引以减少数据处理
- 限制时间范围以避免高容量日志组超时
- 考虑查询的复杂性和执行时间限制

计划规划

- 通过确保在下次计划运行之前完成查询，避免重叠执行
- 设置时间范围时考虑日志摄取延迟
- 在特定时间使用 cron 表达式
- 分散时间表以确保没有达到查询并发限制

监控和维护

- 定期监控执行历史记录以识别故障或性能问题
- 定期审查和更新 IAM 角色以维护安全
- 在部署到生产环境之前测试目标的可访问性

Authorization

- 所有用于计划查询的 API 都对计划查询资源进行授权，而不是对它在输入中占用的资源（如日志组）进行授权。相应地设置 IAM 政策

- 使用 API 中传递的执行角色管理日志组的授权

计划查询入门

创建计划查询时，您将配置多个关键组件，这些组件定义查询的运行方式和结果的交付地点。了解这些组件将帮助您设置有效的自动日志分析。

每个计划查询都包含以下关键组件：

查询配置

CloudWatch Logs Insights 查询字符串、目标日志组和用于分析的查询语言。

计划表达式

用于定义查询何时运行的 cron 表达式或频率日历。您可以指定时区设置以确保查询在正确的本地时间运行。控制台显示了人类可读的日程安排描述，例如 “每周二 15:10 运行查询，时间范围为 5 分钟，在 UTC 时间立即生效，直到无限期。”

时间范围

每次查询执行的回顾周期，由执行时间的开始时间偏移量定义。这决定了每次执行查询将分析多少历史数据。

执行时间表预览

控制台显示接下来的三次计划查询的运行时间以及确切的日期和时间（例如， 2025/10/28 15:10，UTC；/ 2025/1104 15:10，UTC； 2025/11/11 15:10，UTC），帮助您验证计划配置是否正确。

目标

其中，成功执行后会传送查询结果。支持的目标包括 Amazon S3 存储桶，默认情况下，结果元数据将发送到默认事件总线。

执行角色

L CloudWatch ogs 代入的 IAM 角色来执行查询并将结果传送到指定目的地。

在创建计划查询之前，请确保您已配置必要的权限和资源。

创建计划查询

创建计划查询，该查询可自动运行 L CloudWatch ogs Insights 查询并将结果传送到您选择的目的地。

先决条件

在创建计划查询之前，请确保您具备以下条件：

- 日志组 -一个或多个包含要分析的数据的日志组
- 执行 IAM 角色 -具有以下权限的 IAM 角色：
  - `logs:StartQuery`-启动 CloudWatch 日志见解查询的权限
  - `logs:GetQueryResults`-检索查询结果的权限
  - `logs:DescribeLogGroups`-访问日志组信息的权限。只有基于前缀的日志组才需要此项才能进行日志组发现
- 目的地权限 -所选目的地的其他 IAM 权限：
  - 对于亚马逊 S3 目的地：`s3:PutObject`
- For AWS CLI 和 API 用法 -已配置的 AWS 凭据具有调用 CloudWatch 日志 API 的权限

有关详细的 IAM 策略示例，请参阅亚马逊 CloudWatch 日志的身份和访问管理。还需要注意的是，每个账户只能有 1000 个预定查询。

Console

创建计划查询（控制台）

1. 打开 CloudWatch 日志控制台，网址为 https://us-east-1.console.aws.amazon.com/cloudwatch/home ? region=us-east-1 #logs。V2:logs-insights
2. 在导航窗格中，选择 L ogs Insights 。
3. 选择 “创建预设查询” 。
4. 在 “查询定义” 部分：
   a. 对于查询语言，请从列表中选择要使用的查询语言。
   b. 对于查询字符串，在框中输入您的 CloudWatch 日志见解查询。
   c. 对于日志组，从列表中选择要查询的日志组。
5. 在 “时间表设置” 部分：
   a. 对于计划表达式，配置查询的运行时间。从预定义选项中选择或输入自定义 cron 表达式。
   b. 要在创建时生效，请指定计划何时生效。使用 YYYY/MM/DD 格式选择立即开始或在特定日期和时间开始。

c. 对于时间范围，指定每次查询执行的回顾周期。输入持续时间（以分钟为单位），该持续时间定义查询的执行时间有多远。

d. 对于无限期继续，请指定计划何时结束。使用 YYYY/MM/DD 格式选择无限期运行或直到特定的日期和时间。

6. 控制台根据您的配置显示接下来的三次计划查询运行，以 UTC 显示执行查询的确切日期和时间。

7. 在 “将查询结果发布到 S3-可选” 部分（如果使用 S3 目标）中：

a. 对于 S3 存储桶，如果目标存储桶在同一个账户中，则选择此 AWS 账户；如果存储桶位于不同的账户中，则选择另一个 AWS 账户，并提供存储桶拥有账户的账户 ID 作为输入。

b. 对于 Amazon S3 URI，输入用于存储结果的 Amazon S3 存储桶和前缀（例如，`s3://my-bucket/query-results/`）。如果您选择了此账户，则可以选择 “浏览 Amazon S3” 来浏览和选择现有的 Amazon S3 位置。

c. （可选）对于 KMS 密钥 ARN，输入客户管理 AWS KMS 密钥的 ARN 以加密查询结果。SSE-KMS密钥必须与目标 Amazon S3 存储桶位于同一 AWS 区域。

8. 在 “用于将查询结果发布到 Amazon S3 的 IA M 角色” 部分中，选择以下选项之一：

a. 选择Auto-create 一个具有默认权限的新角色来自动设置一个 IAM 角色，该角色具有 CloudWatch 日志向 Amazon S3 传送查询结果所需的权限。

b. 选择 “使用现有角色” 选择具有 CloudWatch 日志向 Amazon S3 传送查询结果所需策略的现有 IAM 角色。使用搜索字段从列表中查找并选择相应的 IAM 角色。

> **Note**
>
> 如果您的账户中启用了角色管理器，则会为您 CloudWatch附加角色，此处描述的角色选项（例如 “创建新角色” 按钮）将替换为 “自定义” 选项。要使用其它角色，请选择自定义。有关创建 IAM 角色的更多信息，请参阅 IAM 用户指南中的 IAM 角色创建。

9. 在 “计划查询执行的 IAM 角色” 部分中，选择以下选项之一：

a. 选择Auto-create 具有默认权限的新角色以自动设置一个 IAM 角色，该角色具有 CloudWatch 日志执行计划查询所需的权限。

b. 选择使用现有角色选择具有 CloudWatch 日志执行计划查询所需策略的现有 IAM 角色。使用搜索字段从列表中查找并选择相应的 IAM 角色。

> **Note**
>
> 如果您的账户中启用了角色管理器，则会为您 CloudWatch附加角色，此处描述的角色选项（例如“创建新角色”按钮）将替换为“自定义”选项。要使用其它角色，请选择自定义。有关创建 IAM 角色的更多信息，请参阅 IAM 用户指南中的 IAM 角色创建。

10. 选择“创建计划”以创建计划查询。

AWS CLI

要创建计划查询 (AWS CLI)

- 使用`create-scheduled-query`命令创建新的计划查询：

```
aws logs create-scheduled-query \
    --name "ErrorAnalysisQuery" \
    --query-language "CWLI" \
    --query-string "fields @timestamp, @message | filter @message like /ERROR/ |
 stats count() by bin(5m)" \
    --schedule-expression "cron(8 * * * ? *)" \
    --execution-role-arn "arn:aws:iam::123456789012:role/
CloudWatchLogsScheduledQueryRole" \
    --log-group-identifiers "/aws/lambda/my-function" "/aws/apigateway/my-api" \
    --state "ENABLED"
```

API

创建计划查询 (API)

- 使用`CreateScheduledQuery`操作创建新的计划查询。以下示例创建了每小时运行一次的计划查询：

```
{
    "name": "ErrorAnalysisQuery",
    "queryLanguage": "CWLI",
    "queryString": "fields @timestamp, @message | filter @message like /ERROR/ |
 stats count() by bin(5m)",
    "scheduleExpression": "cron(8 * * * ? *)",
```

```
    "executionRoleArn": "arn:aws:iam::123456789012:role/
CloudWatchLogsScheduledQueryRole",
    "logGroupIdentifiers": ["/aws/lambda/my-function", "/aws/apigateway/my-
api"],
    "state": "ENABLED"
}
```

创建计划查询后，您可以通过 “计划查询” 页面和 API 来查看和管理该查询， ListScheduledQueries API 会显示所有计划查询及其名称、创建日期、上次运行状态、上次触发时间和重复频率。

查看和管理预定查询

每个查询都有以下信息：

Name

您为计划查询分配的唯一名称。选择名称以查看详细的配置和执行历史记录。

创建日期

计划查询的创建日期，以 YYYY-MM-DD格式显示。

上次运行的状态

最近运行的查询的执行状态。可能的值包括：

- 完成 -查询成功执行，结果已传送到所有配置的目的地。
- 失败 -查询执行或结果传送失败。查看执行历史记录以了解错误详情。
- 无效查询 -该查询无效且存在语法问题
- 超时 -查询已超时。60 分钟后查询会自动超时

上次触发时间

上次执行查询的日期和时间，以 YYYY-MM-DD HH:MM:SS 格式显示。如果查询尚未运行，则显示 “从不”。

重复每次

查询的计划频率。对于使用 cron 表达式的查询，显示自定义，或显示特定频率描述以简化计划。

计划查询页面概述了所有计划查询，显示了它们的当前状态和执行历史记录，以便您可以从一个集中的位置查看、监控和管理所有计划查询。使用这些信息来监控查询性能、识别问题并管理您的自动日志分析工作流程。

Console

查看预设查询（控制台）

1. 打开 CloudWatch 日志控制台，网址为 https://us-east-1.console.aws.amazon.com/cloudwatch/home？region=us-east-1 #logs。V2:logs-insights
2. 在 CloudWatch 日志控制台中，选择计划查询，查看计划查询。

AWS CLI

列出预定查询 (AWS CLI)

- 使用`list-scheduled-queries`命令列出所有计划查询：

```
aws logs list-scheduled-queries --max-results 10
```

API

列出预定查询 (API)

- 使用`ListScheduledQueries`操作检索所有计划查询：

```
{
    "maxResults": 10
}
```

计划查询页面标题显示您账户中计划查询的总数，可帮助您跟踪使用情况并有效地管理自动日志分析工作流程。

查看预定查询执行历史记录

使用执行历史记录来监控计划查询的性能，并解决与查询执行或结果交付相关的任何问题。

执行历史记录显示每次查询运行的状态，包括成功执行、失败和目标处理结果。您可以使用此信息来识别模式、诊断问题并验证您的查询是否按预期运行。

Console

查看执行历史记录（控制台）

1. 在 CloudWatch 日志控制台中，选择计划查询，查看计划查询。
2. 选择要检查的计划查询。
3. 选择 Execution history (执行历史记录) 选项卡。

AWS CLI

查看执行历史记录 (AWS CLI)

1. 使用`get-scheduled-query-history`命令检索计划查询的执行历史记录：

```
aws logs get-scheduled-query-history \
    --identifier "DailyErrorMonitoring" \
    --start-time 1743379200 \
    --end-time 1743465600 \
    --max-results 10
```

2. 要按执行状态过滤，请添加`--execution-statuses`参数：

```
aws logs get-scheduled-query-history \
    --identifier "DailyErrorMonitoring" \
    --start-time 1743379200 \
    --end-time 1743465600 \
    --max-results 1 \
    --execution-statuses "SUCCEEDED"
```

API

查看执行历史记录 (API)

- 使用`GetScheduledQueryHistory`操作检索执行历史记录：

```
{
    "identifier": "DailyErrorMonitoring",
    "startTime": 1743379200,
    "endTime": 1743465600,
    "maxResults": 10,
```

```
    "executionStatuses": ["SUCCEEDED", "FAILED"]
}
```

执行历史显示：

- 执行状态 -正在运行、完成、失败、超时或 InvalidQuery
- 触发时间 -执行查询的时间
- 目标 -每个已配置目标的处理状态 EventBridge，包括 S3 和查询表
- 错误消息 -有关查询执行或目标处理中任何失败的详细信息

更新预定查询

修改您的预定查询配置，以根据需求的变化更改查询字符串、计划、目标或执行角色。

您可以更新计划查询的任何方面，包括查询字符串、计划表达式、目标和执行角色。更改立即生效，以备将来执行。

Console

更新计划查询（控制台）

1. 在 CloudWatch 日志控制台中，选择计划查询，查看计划查询。
2. 选择要更新的计划查询。
3. 选择编辑。
4. 根据需要修改配置。
5. 选择保存更改。

AWS CLI

更新预定查询 (AWS CLI)

- 使用`update-scheduled-query`命令修改现有的计划查询：

```
aws logs update-scheduled-query \
    --identifier "arn:aws:logs:us-east-1:111122223333:scheduled-
query:5e0c0228-1c29-4d26-904f-59f1f1ba3c8f" \
    --description "Monitor for ERROR level logs daily" \
```

```
    --query-language "LogsQL" \
    --query-string "fields @timestamp, @message | filter @message like /ERROR/"
 \
    --log-group-identifiers "/aws/lambda/my-function-1" "/aws/lambda/my-
function-2"
```

API

更新计划查询 (API)

1. 使用UpdateScheduledQuery操作修改预设查询配置：

```
{
    "identifier": "arn:aws:logs:us-east-1:111122223333:scheduled-
query:5e0c0228-1c29-4d26-904f-59f1f1ba3c8f",
    "queryString": "fields @timestamp, @message | filter @message like /WARNING|
ERROR/ | stats count() by bin(5m)",
    "scheduleExpression": "cron(0 */2 * * ? *)",
    "state": "ENABLED"
}
```

2. 要同时更新多个配置参数，请执行以下操作：

```
{
    "identifier": "arn:aws:logs:us-east-1:111122223333:scheduled-
query:5e0c0228-1c29-4d26-904f-59f1f1ba3c8f",
    "queryString": "fields @timestamp, @message, @level | filter @level =
 'ERROR'",
    "scheduleExpression": "cron(0 8,12,16 * * ? *)",
    "executionRoleArn": "arn:aws:iam::111122223333:role/
UpdatedScheduledQueryRole",
    "logGroupIdentifiers": ["/aws/lambda/my-function", "/aws/lambda/another-
function"],
    "destinationConfiguration": {
        "s3Configuration": {
            "destinationIdentifier": "s3://111122223333-sqn-results-bucket/
processed-results",
            "roleArn": "arn:aws:iam::111122223333:role/Admin"
        }
    }
}
```

为计划查询配置 S3 目标

将 Amazon S3 配置为目的地，将您的预定查询结果存储为 JSON 文件，以便长期保留和分析。

使用 Amazon S3 作为目标时，查询结果以 JSON 文件形式存储在指定的存储桶和前缀中。此选项非常适合存档结果、执行批量分析或与其他处理 S3 数据的 AWS 服务集成。

您可以将查询结果传送到计划查询所在 AWS 账户中的 Amazon S3 存储桶，也可以将查询结果传送到其他 AWS 账户中的存储桶。您还可以选择使用客户管理的 AWS KMS 密钥 (SSE-KMS) 对查询结果进行加密。

将结果传送到同一账户中的 Amazon S3 存储桶

当目标 Amazon S3 存储桶与计划查询位于同一个 AWS 账户中时，您可以直接从控制台浏览和选择存储桶。

配置同一个账户的 Amazon S3 目标（控制台）

1. 在 “将查询结果发布到 S3” 部分中，对于 S3 存储桶，选择此账户。
2. 对于 Amazon S3 URI，输入用于存储结果的 Amazon S3 存储桶和前缀（例如，`s3://my-bucket/query-results/`），或者选择 B rowse Amazon S3 浏览并选择现有的 Amazon S3 位置。
3. （可选）要使用客户管理的 AWS KMS 密钥加密结果，请在 KMS AWS KMS 密钥 ARN 字段中输入密钥的 ARN 。密钥必须与目标 Amazon S3 存储桶位于同一 AWS 区域。如果您未指定 AWS KMS 密钥，则存储桶的默认加密设置适用。
4. 在 “用于向 Amazon S3 发布查询结果的 IAM 角色” 部分中，选择Auto-create 具有默认权限的新角色以自动设置所需权限，或选择使用现有角色选择具有所需策略的现有 IAM 角色。

目标交付 IAM 角色需要以下权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-bucket/prefix/*",
            "Condition": {
```

```
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            }
        }
    ]
}
```

将结果传送到另一个账户中的 Amazon S3 存储桶

您可以将预定查询结果传送到其他 AWS 账户中的 Amazon S3 存储桶。使用跨账户存储桶时，您必须提供 Amazon S3 URI 和存储桶拥有账户的账户 ID。

配置跨账户 Amazon S3 目标（控制台）

1. 在“将查询结果发布到 S3”部分中，对于 S3 存储桶，选择其他账户，并提供存储桶拥有账户的账户 ID 作为输入。
2. 对于亚马逊 S3 URI，输入目标存储桶的完整 Amazon S3 URI，并在其他账户中输入前缀（例如，`s3://cross-account-bucket/query-results/`）。
3. （可选）要使用客户管理的 AWS KMS 密钥加密结果，请在 KMS AWS KMS 密钥 ARN 字段中输入密钥的 ARN 。密钥必须与目标 Amazon S3 存储桶位于同一 AWS 区域。
4. 在“用于向 Amazon S3 发布查询结果的 IAM 角色”部分中，选择Auto-create 具有默认权限的新角色以自动设置所需权限，或选择使用现有角色选择具有所需策略的现有 IAM 角色。

Cross-account 交付需要双方的许可。源账户中的目标交付 IAM 角色需要以下权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::cross-account-bucket/prefix/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "123456789012"
                }
            }
        }
    ]
```

```
}
```

目标账户中的 Amazon S3 存储桶策略必须向源账户的 IAM 角色授予写入对象的权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowScheduledQueryRolePutObject",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/my-s3-delivery-role"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::cross-account-bucket/prefix/*"
        }
    ]
}
```

由客户管理对结果进行加密 AWS KMS 键

您可以选择指定客户管理的 AWS KMS 密钥来加密交付给 Amazon S3 的查询结果 SSE-KMS。 AWS KMS 密钥可以与计划查询位于同一个账户中，也可以放在不同的账户中。

当您指定 AWS KMS 密钥时，计划查询使用该密钥通过加密结果 SSE-KMS。当您未指定 AWS KMS 密钥时，存储桶的默认加密设置将适用。如果使用客户管理的密钥将存储桶配置为默认 SSE-KMS 加密，则目标交付 IAM 角色仍必须拥有该密钥的`kms:GenerateDataKey`权限。

目标交付 IAM 角色需要 AWS KMS 密钥`kms:GenerateDataKey`权限。以下示例显示了使用客户管理 AWS KMS 密钥的 Amazon S3 目标所需的权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-bucket/prefix/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
```

```
            }
        },
        {
            "Effect": "Allow",
            "Action": "kms:GenerateDataKey",
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.us-east-1.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::my-bucket*"
                }
            }
        }
    ]
}
```

当 AWS KMS 密钥与目标交付 IAM 角色不同的账户中时，AWS KMS 密钥拥有账户中的密钥策略必须明确授予该角色访问权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowScheduledQueryRoleToEncrypt",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/my-s3-delivery-role"
            },
            "Action": "kms:GenerateDataKey",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.us-east-1.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::my-bucket*"
                }
            }
        }
    ]
}
```

> **Note**
>
> 当 AWS KMS 密钥和目标交付 IAM 角色位于同一个账户中时，如果 AWS KMS 密钥策略包含默认的 “启用 IAM 策略” 根语句，则仅 IAM 身份策略就足够了。仅当 AWS KMS 密钥策略未委托给 IAM 时，才需要明确的密钥策略授予。

用于将查询结果发布到 Amazon S3 的 IAM 角色必须与 IAM 角色分开配置，以便按计划执行查询。这种分离允许细粒度的访问控制，执行角色可以运行查询，而 Amazon S3 角色专门处理结果交付。这两个角色都必须包含允许 CloudWatch 日志服务 (`logs.amazonaws.com`) 代入该角色的信任策略。

为计划查询配置查找表目的地

将查找表配置为目的地，使用计划查询结果自动刷新查找表。在每次计划执行时，L CloudWatch ogs 都会使用查询结果填充指定的查询表，从而使用最新的日志数据使该表保持最新状态。然后，您可以使用其他日志查询中的`lookup`命令引用该表。有关查找表的更多信息，请参阅lookup。

如果查找表不存在，L CloudWatch ogs 会在第一次成功执行时创建它。后续的每次执行都是完全的替换操作——所有现有的表格内容都将替换为新的查询结果。

查找表配置包括以下设置：

- 表名 -要创建或刷新的查找表的名称。该名称只能包含字母数字字符和下划线。
- 交付角色 — IAM 角色的 ARN，该角色授予 CloudWatch 日志创建或更新包含查询结果的查询表的权限。
- 描述 -（可选）对查询表的描述。
- AWS KMS key —（可选）用于加密查询表数据的 AWS KMS 密钥的 ARN。如果您未指定密钥，则数据将使用 AWS-owned 密钥加密。
- 标签 -（可选）与查找表关联的配 Key-value 对。标签仅在最初创建表时应用。

目标交付 IAM 角色需要以下权限，并且必须包含允许 CloudWatch 日志服务 (`logs.amazonaws.com`) 代入该角色的信任策略：

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

```
            "Action": [
                "logs:CreateLookupTable",
                "logs:UpdateLookupTable",
                "logs:GetQueryResults"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 查找表目标结果受与您直接创建的表相同的查询表配额的限制，例如每个账户的最大查找表数量 AWS 区域。如果计划执行无法刷新表，则现有表内容保持不变，并且失败会出现在计划查询执行历史记录中。

对计划查询进行故障排除

使用这些疑难解答主题来解决计划查询的常见问题。

查询执行因权限错误而失败

解决导致计划查询无法执行或将结果传送到目标的权限错误。

当执行角色缺乏从日志组读取或写入目标资源所需的权限时，就会出现权限错误。

解决权限错误

1. 验证执行角色对目标日志组具有`logs:StartQuery`、`logs:GetQueryResults`和`logs:DescribeLogGroups`权限。
2. 确保执行角色拥有目标资源（`s3:PutObject`例如 S3 存储桶）的写入权限。
3. 检查信任策略是否允许 CloudWatch Logs 担任执行角色。该角色应信任其信任策略中的日志服务主体 (`logs.amazonaws.com`)。

常见原因包括缺少 IAM 权限、策略中不正确的资源 ARN 或信任策略配置问题。

为防止权限错误，请在创建执行角色时使用最小权限原则，并在将计划查询部署到生产环境之前测试权限。

查询超时

解决计划查询超过最大执行时间限制时发生的超时错误。

当查询处理指定数据范围所需的时间超过 60 分钟时，就会发生查询超时，这通常是由于大型数据集或复杂的查询逻辑所致。

解决超时错误

1. 通过减少开始时间偏移量来缩短时间范围，从而减少每次执行处理的数据。
2. 通过在查询的早期添加过滤器来优化查询，以减少处理的数据量。使用筛选器索引来减小数据扫描大小。
3. 考虑将复杂的查询分解为更简单、更有针对性的查询。

常见原因包括查询较长时间范围、处理高容量日志组或在未进行适当筛选的情况下使用复杂的聚合。

为防止超时，请在L CloudWatch ogs Insights中使用预期的数据量手动测试查询，并在计划之前优化性能。

目标处理失败

解决无法将计划查询结果传送到配置的目的地时发生的故障。

当目标 Amazon S3 存储桶或 EventBridge 事件总线无法访问或配置不正确时，就会发生目标处理故障。

解决查询结果未发布到目标的故障

1. 验证指定的 Amazon S3 存储桶是否存在且可访问。
2. 检查目标配置中是否有正确的 URI。
3. 确保执行角色具有写入目标所需的权限。

常见原因包括删除或重命名的目标资源、不正确的目标 URI 或网络连接问题。

为防止目标故障，请定期验证目标配置并监控目标资源的可用性。

无效的查询错误

解决计划查询字符串中阻碍成功执行的语法和逻辑错误。

当查询字符串包含语法错误、引用不存在的字段或使用不支持的查询语言功能时，会出现无效的查询错误。

解决无效的查询错误

1. 在 L CloudWatch ogs Insights 中手动测试您的查询，以验证语法和逻辑。
2. 检查目标日志组中是否存在所有引用的日志字段。
3. 确认计划查询支持您使用的查询语言功能。

常见原因包括字段名称中的拼写错误、不正确的查询语法或使用计划执行环境中不支持的查询功能。

为防止无效的查询错误，请务必在安排查询之前进行交互式测试，并使用字段发现功能来验证字段名称。

查询并发错误

由于计划查询使用的配额与 Cloudwatch Logs 见解查询的配额相同，因此出现并发错误时，需要记住以下几点。建议分散日程安排，以避免达到并发限制。

- 配额：每个 AWS 账户最多可以运行 100 个并发 CloudWatch 日志见解查询。
- 仪表板：添加到 CloudWatch 仪表板的查询也计入此并发限制，因为它们是在仪表板加载或刷新时执行的。
- OpenSearch 服务 PPL/SQL：每个 AWS 账户最多可以运行 15 个并发 OpenSearch PPL 或 OpenSearch SQL 查询。
- Cross-account 查询：并发配额适用于单账户和跨账户查询。使用 CloudWatch 跨账户可观察性时，在监控账户中针对关联源账户发起的查询也计入监控账户的并发限制。
- 不频繁访问日志组：对于不频繁访问日志类中的日志组，并发 Logs Insights 查询的最大数量限制为五个。

日志异常检测

您可以通过两种方式检测日志数据中的异常：创建用于持续监控的日志异常检测器，或者使用 Logs Insights 查询中的anomaly detection CloudWatch 命令进行按需分析。

日志异常检测器会扫描摄取到日志组中的日志事件，并自动查找日志数据中的异常。异常检测使用机器学习和模式识别来建立典型日志内容的基准。要进行按需分析，您可以使用 Logs Insigh CloudWatch ts 查询中的anomaly detection命令来识别时间序列数据中的异常模式。有关基于查询的异常检测的更多信息，请参阅在 “ CloudWatch 日志见解” 中使用异常检测。

在为日志组创建异常检测器后，它使用日志组中过去两周的日志事件进行训练。训练时间可能长达 15 分钟。训练完成后，它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压缩成几个模式。

例如，请参阅以下三个日志事件示例。

```
2023-01-01 19:00:01 [INFO] Calling DynamoDB to store for ResourceID: 12342342k124-12345
2023-01-01 19:00:02 [INFO] Calling DynamoDB to store for ResourceID: 324892398123-1234R
2023-01-01 19:00:03 [INFO] Calling DynamoDB to store for ResourceID: 3ff231242342-12345
```

在上面的示例中，所有三个日志事件都遵循一种模式：

```
<Date-1> <Time-2> [INFO] Calling DynamoDB to store for resource id <ResourceID-3>
```

模式内的字段称为令牌。模式内变化的字段（如请求 ID 或时间戳）称为动态令牌。为动态令牌找到的每个不同值称为令牌值。

如果 CloudWatch Logs 可以推断出动态令牌所代表的数据类型，则它会将该令牌显示为<*string*-*number*>。*string*是对令牌所代表的数据类型的描述。*number*显示了与其他动态代币相比，该代币在模式中的显示位置。

CloudWatch Logs 根据对包含该名称的日志事件内容的分析来分配名称的字符串部分。

如果 CloudWatch Logs 无法推断出动态令牌所代表的数据类型，则它会将该令牌显示为 <Token-*number* >，并*number*指明与其他动态令牌相比，该令牌在模式中的显示位置。

动态令牌的常见示例包括错误代码、IP 地址、时间戳和请求 IDs。

日志异常检测使用这些模式来查找异常。异常检测器模型训练期结束后，将根据已知趋势对日志进行评估。异常检测器将显著的波动标记为异常。

本章介绍了如何启用异常检测、查看异常、为日志异常检测器创建警报以及日志异常检测器发布的指标。它还描述了如何对异常检测器及其结果进行 AWS Key Management Service加密。

创建日志异常检测器不会产生费用。

异常和模式的严重性和优先级

日志异常检测器发现的每个异常都分配有优先级。发现的每种模式都分配有严重性。

- 优先级是自动计算的，它基于模式的严重性级别以及与预期值的偏差量。例如，如果某个令牌值突然增加 500%，则即使异常的严重性为 NONE，也可能会被指定为 HIGH 优先级。
- 严重性仅基于模式中找到的关键字，例如 FATAL、ERROR 和 WARN。如果未找到这些关键字，则模式的严重性将标记为 NONE。

异常可见性时间

创建异常检测器时，请为其指定最长异常可见性期限。这是异常在控制台中显示并由 ListAnomaliesAPI 操作返回的天数。在异常经过此时间段后，如果异常继续发生，则会自动接受为正常行为，并且异常检测器模型会停止将其标记为异常。

如果在创建异常检测器时不调整可见性时间，则默认使用 21 天。

抑制异常

发现异常后，您可以选择暂时或永久抑制它。抑制异常会导致异常检测器在您指定的时间内停止将此事件标记为异常。抑制异常时，您可以选择仅抑制该特定异常，或抑制与发现异常的模式相关的所有异常。

您仍然可以在控制台中查看被抑制的异常。您也可以选择停止抑制它们。

常见问题

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合进行异常检测？

日志异常检测适用于：应用程序日志和其他类型的日志，其中大多数日志条目符合典型模式。事件包含日志级别或严重性关键字（如 INFO、ERROR 和 DEBUG）的日志组尤其适合日志异常检测。

日志异常检测不适合：记录具有极长 JSON 结构的事件，例如 CloudTrail 日志。模式分析仅分析日志行的前 1500 个字符，因此超出该限制的任何字符都将被跳过。

审计或访问日志（如 VPC 流日志）在异常检测方面的成功率也较低。异常检测旨在发现应用程序问题，因此它可能不太适合网络或访问异常。

为了帮助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，则异常检测可能会效果良好。有关模式分析的更多信息，请参阅 模式分析。

什么会被标记为异常？

以下情况可能会导致日志事件被标记为异常：

- 具有日志组中以前未见过的模式的日志事件。
- 已知模式的显著变化。
- 具有一组离散的常用值的动态令牌的新值。
- 动态令牌的值出现次数的大幅变化。

虽然上述所有项都可能被标记为异常，但它们并不一定意味着应用程序性能不佳。例如， higher-than-usual许多200成功值可能会被标记为异常。在这种情况下，您可以考虑抑制这些不表示问题的异常。

被屏蔽的敏感数据会发生什么情况？

日志事件中被屏蔽为敏感数据的任何部分都不会进行异常扫描。有关屏蔽敏感数据的更多信息，请参阅通过屏蔽帮助保护敏感的日志数据。

在 “ CloudWatch 日志见解” 中使用异常检测

除了创建用于持续监控的日志异常检测器外，您还可以在 Lo CloudWatch gs Insights 查询中使用anomaly命令按需识别日志数据中的异常模式。此命令扩展了现有 `pattern` 功能，并使用机器学习来检测五种类型的异常，包括模式频率变化、新模式和令牌变化。

`anomaly` 命令尤其适用于：

- 对历史日志数据进行临时分析，以识别异常模式

- 调查特定时间段内的异常行为
- 监控 Lambda 函数等应用程序的执行问题

有关在查询中使用 `anomaly` 命令的更多信息，请参阅 异常。

这种基于查询的异常检测是对以下各节中所述的连续异常检测器的补充，为您提供实时监控和按需分析功能。

对日志组启用异常检测

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。

您还可以以编程方式创建异常检测器。有关更多信息，请参阅 CreateLogAnomalyDetector。

创建日志异常检测器

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 选择日志、日志异常。
3. 选择创建异常检测器。
4. 选择要为其创建此异常检测器的日志组。
5. 在异常检测器名称中输入检测器的名称。
6. （可选）将评估频率从默认 5 分钟更改为其他频率。根据日志组接收新日志的频率设置此值。例如，如果日志组每 10 分钟批量接收新的日志事件，则将评估频率设置为 15 分钟可能合适。
7. （可选）要将异常检测器配置为仅在包含特定单词或字符串的日志事件中查找异常，请选择筛选模式。

   然后，在异常检测筛选模式中输入模式。有关模式语法的更多信息，请参阅 指标筛选条件、订阅筛选条件、筛选日志事件和 Live Tail 的筛选条件模式语法。

   （可选）要测试筛选模式，请在日志事件消息中输入一些日志消息，然后选择测试模式。
8. （可选）要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. 要更改默认异常可见性期限，请在最长异常可见性期限（天）中输入一个新值。

   b. 要将 AWS KMS 密钥与此异常检测器关联，请在 KMS 密钥 ARN 中输入 ARN。如果您分配了一个密钥，则此检测器发现的异常信息将使用该密钥进行静态加密。用户必须拥有此密钥的权限以及异常检测器检索其发现的异常相关信息的权限。

您还必须确保 CloudWatch 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。

   根据日志组正在提取的日志事件，将创建异常检测器并开始训练其模型。大约 15 分钟后，异常检测开始启动并开始查找和发现异常。

查看已发现的异常

创建一个或多个日志异常检测器后，您可以使用 CloudWatch 控制台查看他们发现的异常。

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、日志异常。

   将出现日志异常表。日志异常旁边顶部的数字显示表中列出了多少个日志异常。表中的每一行显示以下信息：

   - 异常列显示异常的简短摘要。这些摘要由 CloudWatch 日志生成。
   - 异常的优先级。优先级是根据日志事件的变化量、关键字（例如日志事件中发生的 `Exception`）等自动计算的。
   - 异常基于的日志模式。有关模式的更多信息，请参阅 日志异常检测。
   - 异常日志趋势显示一个直方图，描述与模式匹配的日志量。
   - 上次检测时间显示最近一次发现此异常的时间。
   - 首次检测时间显示首次发现此异常的时间。
   - 异常检测器显示包含与此异常相关的日志事件的日志组的名称。您可以选择此名称来查看日志组详细信息页面。
3. 要进一步检查一个异常，请选择其行中的单选按钮。

   将出现模式检查窗格并显示以下内容：

   - 此异常基于的模式。选择模式内的一个令牌来分析该令牌的值。
   - 直方图，显示查询的时间范围内异常的出现次数。

- 日志示例选项卡显示一些属于异常一部分的日志事件。
- 如果您已选择动态令牌，则令牌值选项卡会显示所选动态令牌的值。

> **Note**
>
> 每个令牌最多可捕获 10 个令牌值。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，请选择其行中的单选按钮，然后执行以下操作：

   a. 选择操作、抑制异常。

   b. 然后指定希望抑制异常的时长。

   c. 要隐藏与此模式相关的所有异常，请选择抑制模式。

   d. 选择抑制异常。

查看单个日志组中发现的异常

1. 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个日志异常。表中的每一行显示以下信息：

   - 异常列显示异常的简短摘要。这些摘要由 CloudWatch 日志生成。
   - 异常的优先级。优先级是根据日志事件的变化量、关键字（例如日志事件中发生的 `Exception`）等自动计算的。
   - 异常基于的日志模式。有关模式的更多信息，请参阅 日志异常检测。
   - 异常日志趋势显示一个直方图，描述与模式匹配的日志量。
   - 上次检测时间显示最近一次发现此异常的时间。
   - 首次检测时间显示首次发现此异常的时间。
4. 要进一步检查一个异常，请选择其行中的单选按钮。

   将出现模式检查窗格并显示以下内容：

   - 此异常基于的模式。选择模式内的一个令牌来分析该令牌的值。

- 直方图，显示查询的时间范围内异常的出现次数。
- 日志示例选项卡显示一些属于异常一部分的日志事件。
- 如果您已选择动态令牌，则令牌值选项卡会显示所选动态令牌的值。

> **Note**
>
> 每个令牌最多可捕获 10 个令牌值。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，请选择其行中的单选按钮，然后执行以下操作：

   a. 选择操作、抑制异常。

   b. 然后指定希望抑制异常的时长。

   c. 要隐藏与此模式相关的所有异常，请选择抑制模式。

   d. 选择抑制异常。

在日志异常检测器上创建警报

您可以为日志组中的日志异常检测器创建警报。您可以指定当指定时间段内在日志组中发现指定数量的异常时，警报进入 ALARM 状态。您还可以使用筛选条件，以便警报仅计算指定优先级的异常。

为日志异常检测器创建警报

1. 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、日志异常。

   将显示日志异常检测器表。

3. 选中要设置警报的异常检测器的单选按钮，然后选择创建警报。

   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）要按异常优先级筛选此警报，请执行以下操作之一：
   - 要使警报仅计数高优先级异常，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

   有关优先级级别的更多信息，请参阅 异常和模式的严重性和优先级。

5. 选择对警报使用静态或指标异常检测阈值。此选择决定如何设置警报阈值。静态阈值意味着警报阈值是您选择的静态恒定数字。异常检测阈值意味着它 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。如果异常检测器在周期指定的时间内发现超过此数量的警报，则警报进入 ALARM 状态。
7. 选择其他配置。对于触发警报的数据点数，指定必须有多少个评估期（数据点）处于 ALARM 状态才能触发警报。如果此处的两个值匹配，则会创建一个告警；如果多个连续评估期违例，该告警将变为 ALARM（告警）状态。

   要创建“M（最大为 N）”告警，为第一个值指定的数字应小于为第二个值指定的数字。有关更多信息，请参阅评估告警。
8. 对于 Missing data treatment（缺失数据处理），选择在缺失某些数据点时的告警行为。有关更多信息，请参阅配置 CloudWatch 警报如何处理丢失的数据。
9. 选择下一步。
10. 对于通知，选择添加通知，然后指定当警报转换为 ALARM、OK 或 INSUFFICIENT_DATA 状态时要通知的 Amazon SNS 主题。

    a. （可选）要为相同告警状态或不同告警状态发送多个通知，请选择 Add notification（添加通知）。

    > **Note**
    >
    > 我们建议您将警报设置为在进入数据不足状态以及进入警报状态时采取行动。这是因为连接到数据来源的 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. 在名称和描述下，输入警报的名称和描述，然后选择下一步。名称必须仅包含 UTF-8 字符，并且不能包含 ASCII 控制字符。描述可以包含 markdown 格式，该格式仅显示在 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. 在预览和创建下，确认警报的信息和条件正确，然后选择创建警报。

日志异常检测器发布的指标

CloudWatch 日志将AnomalyCount指标发布到 CloudWatch 指标。此指标会发布到 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。作为替代方案，你可以使用 AWS Key Management Service 这种加密。如果这样做，则使用密 AWS KMS 钥进行加密。通过将 KMS 密钥与异常检测器关联，可以在异常检测器级别启用使用 AWS KMS 进行加密。

> **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。

此示例中的 Condition 部分将 AWS KMS 密钥的使用限制于指定账户，但它可用于任何异常检测器。

JSON

```
{
"Version":"2012-10-17",
"Id": "key-default-1",
"Statement": [
    {
        "Sid": "EnableIAMUserPermissions",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
        },
        "Action": "kms:*",
        "Resource": "*"
    },
    {
        "Sid": "AllowCloudWatchLogsEncryption",
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.us-east-1.amazonaws.com"
        },
        "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:GenerateDataKey*"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "123456789012"
            },
```

```
                "StringLike": {
                    "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-
east-1:123456789012:anomaly-detector:*"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:logs:us-
east-1:123456789012:anomaly-detector:*"
                }
            }
        },
        {
            "Sid": "AllowCloudWatchLogsDescribeKey",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        },
        {
            "Sid": "AllowCloudWatchLogsReEncryption",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws-crypto-ec:aws:logs:arn":
 "arn:aws:logs:us-east-1:123456789012:anomaly-detector:*"
```

```
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:logs:us-
east-1:123456789012:anomaly-detector:*"
                }
            }
        },
        {
            "Sid": "AllowCloudWatchLogsDescribeKeyForReEncryption",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

最后，使用以下put-key-policy命令添加更新的策略：

```
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

在控制台或使用或创建 KMS 密钥时，可以将其与异常检测器关联。 AWS CLI APIs

步骤 4：取消密钥与异常检测器的关联

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

使用 L CloudWatch ogs Live Tai

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。

仅标准日志类中的日志组支持 Live Tail。有关日志类的更多信息，请参阅 日志类。

以下部分介绍了如何在控制台和 AWS CLI中使用 Live Tail。您还可以通过编程方式启动 Live Tail 会话。有关更多信息，请参阅 StartLiveTail。有关 SDK 示例，请参阅使用 SD AWS K 启动 Live Tail 会话。

您也可以在 AWS Toolkit for Visual Studio Code中使用 Live Tail。要从 VS Code 命令面板启动 Live Tail 会话，请参阅 AWS Toolkit for Visual Studio Code 用户指南的 Amazon L CloudWatch ogs Live Tail 部分。

Live Tail 功能适用于所有商业 AWS 区域。它在中国地区或 AWS GovCloud （美国）地区不可用。

> **Note**
>
> StartLiveTailAPI 使用 SDK 主机前缀注入来路由请求。2026 年 4 月 1 日之前发布的 SDK 版本会路由到`streaming-logs.Region.amazonaws.com`，但不支持 VPC 终端节点。2026 年 4 月 1 日当天或之后发布的 SDK 版本会路由到`stream-logs.Region.amazonaws.com`，它支持 VPC 终端节点。要为此 API 设置 VPC 终端节点，请参阅为 CloudWatch 日志创建 VPC 终端节点。

使用 Live Tail 开始直播会话 AWS CLI

该`start-live-tail` AWS CLI 命令为终端中的一个或多个日志组启动 Live Tail 流式传输会话。Live Tail 会话最长可持续三个小时。如果每秒有超过 500 个日志事件与筛选条件匹配，则显示的日志事件是总日志事件的样本，以提供实时的跟踪体验。有关 `start-live-tail` 命令的更多信息，请参阅 start-live-tail

您可以在两种模式下使用 `start-live-tail`：

- print-only – 这是默认模式
- interactive

print-only

在 `print-only` 模式下，日志事件在终端上流式传输。每秒都会在底部添加新事件，从而创建类似于 Linux 上的 `tail -f` 的近乎实时的跟踪体验。

要在 print-only 模式下启动 Live Tail 会话，请输入以下命令。

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-
east-1:111111222222:log-group:my-logs
```

当使用 print-only 模式时，您还可以将其与其他 Linux 命令一起传输以增强其分析能力。以下示例使用 `error` 关键字筛选日志事件，并打印这些事件的第二列和第四列，以帮助您提取特定信息。

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-
east-1:111111222222:log-group:my-logs --mode print-only | grep "error" | awk '{print
 $2, $4}'
```

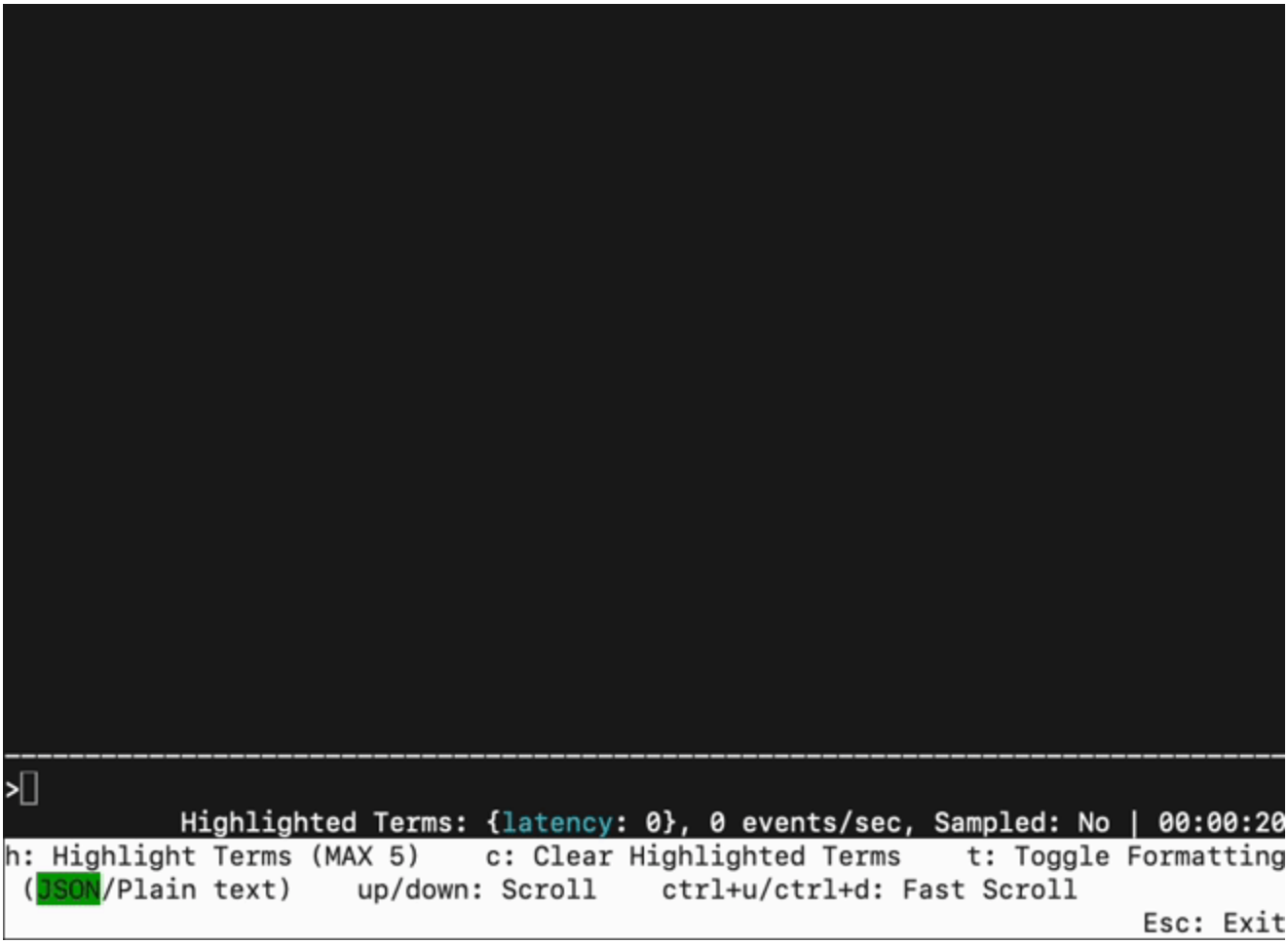
interactive

在 `interactive` 模式下，您可以突出显示术语并在 JSON 和纯文本之间切换输出日志事件的格式。Interactive 模式还显示有关 Live Tail 会话的信息，例如会话持续时间、是否正在对会话进行采样、当前突出显示的术语和遇到这些术语的次数。

要在 interactive 模式下启动 Live Tail 会话，请输入以下命令。

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-
east-1:111111222222:log-group:my-logs --mode interactive
```

Live Tail 会话开始。以下视频展示了示例会话的一部分。



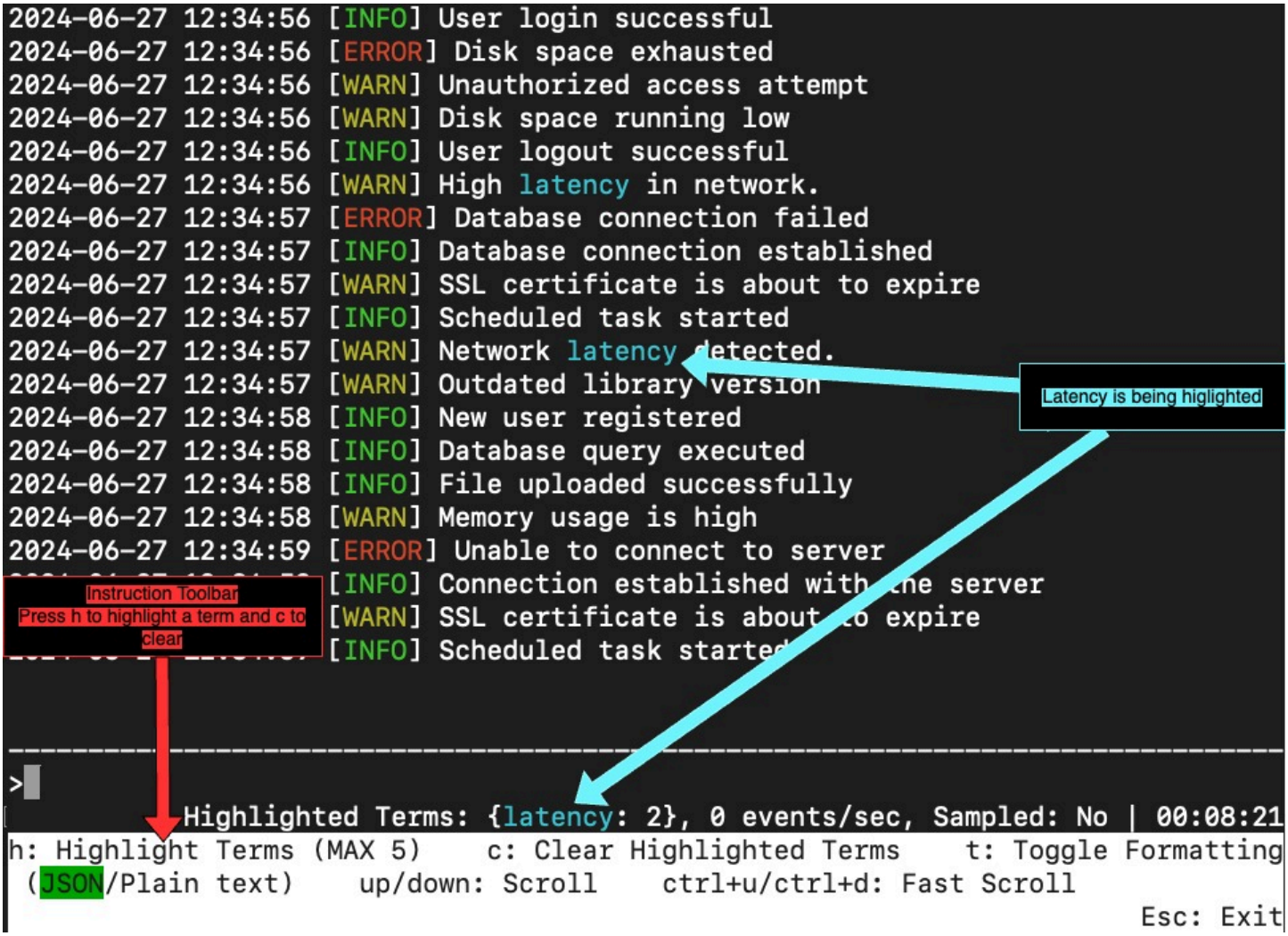
要突出显示流日志中的某个术语，请按 h，然后输入该术语。下面显示了突出显示术语 `latency` 后的屏幕。

要清除突出显示的术语，请按 c，然后键入代表要停止突出显示的术语的数字。

您可以按 t 在 JSON 和纯文本之间切换传入事件的显示格式。此切换功能是尽力而为的，并且仅当日志事件格式兼容时才会发生。

您可以使用向上和向下箭头键滚动，并使用 CTRL+u 和 CTRL+d 更快地滚动。

下图显示了 Live Tail 会话期间 `latency` 术语的突出显示。



在控制台中启动 Live Tail 会话

您可以使用 CloudWatch 控制台启动 Live Tail 会话。以下程序说明了如何使用左侧导航窗格中的 Live tail 选项启动 Live Tail 会话。您也可以从 “日志组” 页面或 “ CloudWatch 日志见解” 页面启动 Live Tail 会话。

如果您使用数据保护策略来屏蔽使用 Live Tail 查看的日志组中的敏感数据，则敏感数据在 Live Tail 会话中始终会被屏蔽。有关屏蔽日志组中敏感数据的详细信息，请参阅 [通过屏蔽帮助保护敏感的日志数据](#)。

> **Important**
>
> 如果您的网络安全团队不允许使用网络套接字，则您目前无法访问 CloudWatch 主机的 Live Tail 部分。你可以将 Live Tail 与 AWS CLI 或 API 一起使用。有关更多信息，请参阅使用 Live Tail 开始直播会话 AWS CLI和StartLiveTail。

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过滤器相匹配。
   - 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. 要暂停事件流以调查当前显示的内容，请在事件窗口中选择任意位置。
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。然后选择开始。

Cross-account 跨区域日志集中化

Amazon L CloudWatch ogs 数据集中化使用跨账户和跨区域集中规则，将来自多个成员账户的日志数据收集到一个数据存储库中。 AWS Organizations 您可以定义自动将日志数据从多个账户和 AWS 区域 复制到组织内集中账户的规则。此功能简化了日志整合，从而改善了整个 AWS 基础架构的集中监控、分析和合规性。

CloudWatch 日志数据集中化提供了配置灵活性，以满足运营和安全要求，例如能够在目标账户中设置规则期间配置备份区域，以确保提高灵活性。此外，您可以完全控制从源账户复制的日志组的加密行为，以处理最初使用客户管理的 KMS 密钥加密的数据。

> **Note**
>
> CloudWatch 日志集中化功能仅处理创建集中规则后到达源账户的新日志数据。历史日志数据（创建规则之前存在的日志）不会进行集中。

数据集中化概念

在开始使用 CloudWatch 日志数据集中化之前，请熟悉以下概念：

集中化规则

一种用于定义如何将源账户和区域中的日志数据复制到目标账户和区域的配置。规则指定源条件和目标设置。

源账户

日志数据来源的 AWS 账户。根据您定义的集中化规则，源账户中的日志事件将复制到目标账户。

目标账户

存储副本日志数据的目标 AWS 帐户。该账户用作日志分析和监控的集中位置。

备份区域

目标账户中的可选辅助区域，可以在其中复制日志数据，以提高恢复能力并实现灾难恢复。

标签传播

一种选择加入功能，可将资源标签从源日志组传播到相应的目标日志组。标签传播使用目标账户中客户管理的 IAM 角色来添加、更新和删除目标日志组上的标签。您可以通过在集中化规则的目标配置中添加`TagPropagationConfiguration`区块来配置标签传播。

CloudWatch 日志中的加密

日志组数据始终在 CloudWatch 日志中加密。默认情况下，CloudWatch 日志使用服务器端加密和 256 位高级加密标准 Galois/Counter 模式 (AES-GCM) 来加密静态日志数据。作为替代方案，您可以使用 AWS 密钥管理服务进行这种加密。有关更多信息，请参阅CloudWatch 日志加密文档。

- 集中化期间加密的工作原理：CloudWatch 日志集中化在摄取时主动将日志数据从源账户复制到目标账户。在此过程中，您的数据在传输过程中将使用 AWS 自有服务密钥进行加密。源日志组和目标日志组中的静态数据均使用您选择的加密方法（客户管理或 AWS 拥有的 KMS 密钥）进行加密。如果您在目标日志组中使用客户管理的 KMS 密钥，请将标签添加`LogsManaged = true`到 kms 密钥中，以便集中化服务访问该密钥。
- 当需要 KMS 权限时：
  - 如果您在源账户中使用客户管理的 KMS 密钥，则在以下示例场景中，CloudWatch 日志需要 KMS 权限：
    - 吞吐量管理：当达到集中吞吐量限制时，日志数据将临时使用客户管理的 KMS 密钥加密存储，直到带宽可用为止。
    - 数据保护和编辑：当源日志组启用了数据保护策略时，CloudWatch 日志需要解密权限才能访问原始日志数据以对其进行集中管理。

> **Important**
>
> 集中化规则由 AWS 组织管理账户或委派管理员管理。要将客户管理的 KMS-encrypted 日志组排除在集中管理范围之外，请将规则设置配置为 “不要集中使用 AWS KMS 密钥加密的日志组”。

设置日志集中化

要设置 CloudWatch 日志集中化，您需要配置集中规则，定义日志数据如何从源账户的日志组流向目标账户中的日志组。

启用集中化规则并将日志事件复制到目标账户后，您可以针对集中式日志组创建指标、订阅和账户筛选条件，并增强筛选功能。这些筛选条件可以针对来自特定源账户和区域的日志事件，并且可以将源账户和区域信息作为指标维度发出。有关更多信息，请参阅 使用筛选条件从日志事件创建指标。

先决条件

- AWS Organizations 必须进行设置，并且源帐户和目标帐户必须都属于组织。

- 必须为管理账户和目标账户启用可信访问权限 CloudWatch，以便提供对日志数据的访问权限。

> **Note**
>
> 建议通过控制台启用可信访问，这会自动创建所需的服务相关角色（SLR）。如果通过其他方法启用可信访问，则需要单独创建服务相关角色。

标签传播的先决条件

要启用标签传播，必须完成以下附加设置：

创建客户管理的 IAM 角色

您必须使用以下配置在目标账户中创建客户管理的 IAM 角色：

信任策略

该角色的信任策略必须允许中心化服务相关角色代入该角色。以下示例信任策略向`AWSServiceRoleForObservabilityAdmin_LogsCentralization`服务相关角色授予代入目标角色的权限。`<destination-account-id>`替换为您的目标账户 ID `<organization-id>` 和您的 AWS Organizations ID。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::<destination-account-id>:role/aws-
service-role/logs-centralization.observabilityadmin.amazonaws.com/
AWSServiceRoleForObservabilityAdmin_LogsCentralization"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "sts:ExternalId": "<organization-id>"
      }
    }
  }]
}
```

权限策略

该角色的权限策略必须授予对目标日志组的标签操作。以下示例授予所需的最低权限。<destination-account-id>用你的价值取代。您可以将范围Resource进一步限定到特定的日志组 ARN 模式。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "logs:ListTagsForResource",
      "logs:TagResource",
      "logs:UntagResource"
    ],
    "Resource": "arn:aws:logs:*:<destination-account-id>:log-group:*"
  }]
}
```

授予 iam: PassRole 权限

您必须拥有目标角色的iam:PassRole权限，通过iam:PassedToService条件密钥将权限范围限定为集中化服务。以下示例授予传递角色的权限。将 <destination-account-id> 和 <your-tag-role-name> 替换为您自己的值。

> **Note**
>
> 此声明适用于您的身份政策（委托人呼叫CreateCentralizationRuleForOrganization或UpdateCentralizationRuleForOrganiz而不是目标角色本身。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::<destination-account-id>:role/<your-tag-role-name>",
    "Condition": {
      "StringEquals": {
        "iam:PassedToService": "logs-centralization.observabilityadmin.amazonaws.com"
```

```
      }
    }
  }]
}
```

自定义目标日志组名称

创建集中化规则时，您可以使用属性自定义目标日志组名称的结构。创建日志组时，这些属性会自动替换为实际值，从而允许您在目标账户中按层次结构组织日志。默认情况下，仅使用该${source.logGroup}属性，它会合并目标账户中所有同名的日志组。如果变量无法解析，它将继承层次结构中其父变量的值。

可用属性

可以在目标日志组名称模式中使用以下属性：

目标日志组名称属性

属性	说明
${source.accountId}	日志来源的 AWS 账户 ID。
${source.region}	日志 AWS 区域 的起源。
${source.logGroup}	源账户的原始日志组名称。
${source.org.id}	您的来源账户的 AWS Organizations ID。
${source.org.ouId}	来源账户的组织单位 ID
${source.org.rootId}	组织根 ID
${source.org.path}	从账户到 root 的完整组织路径

示例

保留原始日志组结构

模式：/centralized/${source.accountId}${source.logGroup}

结果：/centralized/123456789012/aws/lambda/my-function

按账户和地区整理

模式：`/centralized/${source.accountId}/${source.region}`

结果：`/centralized/123456789012/us-east-1`

按组织结构组织

模式：`/logs/${source.org.id}/${source.org.ouId}/${source.accountId}`

结果：`/logs/o-abc123/ou-xyz-12345678/123456789012`

简单的扁平结构

模式：`/centralized-logs`

结果：`/centralized-logs`

最佳实践

- 包括源账户 ID 以轻松识别来自哪个账户日志。
- 如果您要从多个区域进行集中管理，请包括源区域。
- 将目标日志组名称结构化为低于 512 个字符。 CloudWatch 日志强制规定日志组名称的最大长度为 512 个字符。
- 如果启用标签传播，则该模式必须包含所有三个属性：`${source.logGroup}`、`${source.accountId}`、和`${source.region}`。这样可以确保每个源日志组映射到唯一的目标日志组。

创建集中化规则

请按照以下过程创建一个将日志数据从源账户复制到目标账户的集中化规则。

创建集中化规则

1. 在组织的管理或委派管理员帐户中导航到 CloudWatch 控制台。
2. 选择设置。
3. 导航到组织选项卡。
4. 选择配置规则。

5. 通过设置下面的字段来指定源详细信息，然后选择下一步：

   a. 集中化规则名称：为集中化规则输入一个唯一名称。

   b. 源账户：定义源选择标准，以选取要从中集中遥测数据的账户。选择标准可包括：

      - 组织内的成员账户列表
      - 组织内的组织单元列表
      - 整个组织

      您可以通过两种模式提供选择标准：

      - 生成器：基于点击的体验，可生成源选择标准
      - 编辑器：提供源选择标准的自由格式的文本框

      源选择标准支持的语法：

      - 支持的密钥： OrganizationId | OrganizationUnitId | AccountId | *
      - 支持的运算符：= | IN | OR

   c. 源区域：选择一个区域列表以查找要集中的遥测数据。

6. 通过设置下面的字段来指定目标详细信息，然后选择下一步：

   a. 目标账户：在组织中选择一个充当遥测数据中心目标的账户。

   b. 目标区域：选择一个存储集中式遥测数据副本的主要区域。

   c. 备份区域：根据需要选择存储集中式遥测数据的第二个副本的区域。

7. 通过设置下面的字段来指定遥测数据，然后选择下一步：

   a. 日志组：选择以下选项之一：

      - 所有日志组：集中源账户中所有日志组的日志。
      - 筛选日志组：根据选择标准，集中来自源账户中一部分日志组的日志。您可以通过两种模式提供选择标准：
        - Builder：基于选择的体验，用于生成选择标准
        - 编辑器：提供选择条件的自由格式文本框

        您可以使用两个选择标准来筛选日志：

        - 日志组选择标准：指定要集中哪些源日志组的选择标准。

- 支持的密钥： LogGroupName| *
- 支持的运算符：= | != | IN | NOT IN | AND | OR | LIKE | NOT LIKE

- 数据源选择标准：指定要集中哪些数据源的选择标准。
  - 支持的密钥： DataSourceName| DataSourceType
  - 支持的运算符：= | != | IN | NOT IN | AND | OR | LIKE | NOT LIKE

当同时指定了日志组选择标准和数据源选择标准时，日志事件必须符合这两个标准才能集中化。

b. KMS 加密的日志组

> **Important**
>
> CloudWatch 如果集中化规则中提供的 KMS 密钥不允许日志使用日志，则集中化规则将无法将日志从源账户传送到目标 CloudWatch 日志组。如果您在目标日志组中使用客户管理的 KMS 密钥，请在 kms 密钥中添加标签 LogsManaged = true。有关更多信息，请参阅 步骤 2：设置 KMS 密钥的权限。

请选择以下选项之一：

- 使用特定目标客户管理的 KMS 密钥集中使用客户管理的 KMS 密钥加密的源日志组：将使用客户管理的 KMS 密钥加密的源日志组中的日志事件集中到目标账户中使用客户管理的 KMS 密钥加密的目标日志组中。

  选择此设置后，您还必须设置以下内容：

  - 目标加密密钥 ARN：目标账户和主要目标区域中客户管理的 KMS 密钥的 ARN，将与新创建的目标日志组相关联。
  - 备份目标加密密钥 ARN（如果选择了备份区域）：目标账户和备份目标区域中客户管理的 KMS 密钥的 ARN，将与新创建的目标日志组相关联。
  - 跳过集中到未加密的目标日志组（可选）：如果日志组已经存在，但没有客户管理的 KMS 密钥，则 CloudWatch 无法更新其加密。选择此选项可跳过将日志事件从使用客户管理的 KMS 密钥加密的源日志组集中到与客户管理的 KMS 密钥无关的目标日志组中。
- 使用 AWS 自有 KMS 密钥将使用客户管理的 KMS 密钥加密的日志组集中到目标账户中：将使用客户管理的 KMS 密钥加密的源日志组中的日志事件集中到使用 AWS 自有 KMS 密钥加密的新创建的目标日志组中。

- 不要集中使用客户管理的 KMS 密钥加密的日志组：跳过使用客户管理的 KMS 密钥加密的源日志组中的日志事件的集中化。

8. 查看集中化规则，根据需要进行任何最后的编辑，然后选择创建集中化策略。

修改集中化规则

请按照以下过程修改现有的集中化规则。

修改集中化规则

1. 在组织的管理或委派管理员帐户中导航到 CloudWatch 控制台。
2. 选择设置。
3. 导航到组织选项卡。
4. 选择管理规则。
5. 选择要更新规则，然后选择编辑。
6. 根据需要更新规则配置，然后选择下一步继续执行每个步骤。
7. 在步骤 4“查看和配置”中，选择更新集中化策略。

查看集中化规则

请按照以下过程查看现有集中化规则的详细信息。

查看集中化规则

1. 在组织的管理或委派管理员帐户中导航到 CloudWatch 控制台。
2. 选择设置。
3. 导航到组织选项卡。
4. 选择管理规则。
5. 查看所有现有集中化规则的列表，然后选择一个特定规则名称以查看其详细信息。

删除集中化规则

请按照以下过程删除现有的集中化规则。

删除集中化规则

1. 在组织的管理或委派管理员帐户中导航到 CloudWatch 控制台。
2. 选择设置。
3. 导航到组织选项卡。
4. 选择管理规则。
5. 选择要删除的规则，然后选择删除。
6. 确认删除并选择删除。

监控和排查集中化规则

您可以使用 CloudWatch 指标、 CloudWatch 日志控制台和日志来监控集中化规则的状态和 AWS CloudTrail 性能。这有助于确保日志数据成功复制，并识别集中化配置中的任何问题。

CloudWatch 日志提供：

1. 每个集中化规则的规则运行状况
   a. 选择设置。
   b. 导航到组织选项卡。
   c. 选择管理规则。
2. 使用以下方式记录 API 调用 AWS CloudTrail
3. CloudWatch 还发布集中化指标，包括复制的日志事件、错误和限制。有关这些指标及其维度的更多信息，请参阅集中化指标和维度。

集中化规则运行状况

每个集中化规则都有运行状况指示其是否正常运行。您可以通过控制台或使用 API 以编程方式检查规则运行状况。

规则运行状况包括：

- HEALTHY：规则运行正常，并按配置复制日志数据
- UNHEALTHY：规则遇到问题，可能无法正确复制数据
- PROVISIONING：组织的集中化正在设置过程中。

当规则被标记为 UNHEALTHY 时，FailureReason 字段会提供有关需要解决的具体问题的详细信息。

标签传播健康状态

标签传播有自己的运行状况 (TagPropagationStatus)，该状态与日志传输RuleHealth的整体运行状况无关。如果您错误配置了目标角色，则不会影响您的整体规则运行状况——只有标签传播显示为不正常。

- Healthy：最近的标签传播尝试成功。
- Unhealthy：最近的标签传播尝试失败。详情请查看该TagPropagationFailureReason字段。有关补救步骤，请参阅下面的故障排除部分。

使用监控集中化 API 调用 AWS CloudTrail

AWS CloudTrail 记录对集中化服务进行的 API 调用，允许您跟踪配置更改并解决属于您的 AWS Organizations成员的帐户的问题。

集中化的关键 CloudTrail 事件包括：

- CreateCentralizationRuleForOrganization：创建新的集中化规则时
- UpdateCentralizationRuleForOrganization：修改现有规则时
- DeleteCentralizationRuleForOrganization：删除规则时
- GetCentralizationRuleForOrganization：检索规则详细信息时
- ListCentralizationRulesForOrganization：列出规则时

您可以使用 CloudTrail 日志来审核集中配置更改，并将其与性能问题或复制失败关联起来。

监控建议

为确保集中化正常运行，我们建议对提供给 Metrics 的关键集中化指标设置 CloudWatch 警报。CloudWatch 这种主动监控可帮助您及早发现问题，并在整个组织中保持可靠的日志集中化。

需要监控的关键指标包括：

- IncomingCopiedBytes：监控成功复制到目标账户的日志数据量。该指标的突然下降或缺失可能表明存在集中化问题。
- CentralizationError：为集中化过程中的任何错误设置警报，以快速识别和解决问题。

- `CentralizationThrottled`：监控可能影响日志复制性能的限制事件。

有关可用集中化指标及其维度的完整列表，请参阅集中化指标和维度。

如果日志未按预期进行集中化，请查看以下可能阻止日志集中化的常见场景。

历史日志数据

CloudWatch 日志集中化功能仅处理创建集中规则后到达源账户的新日志数据。历史日志数据（创建规则之前存在的日志）不会进行集中。

KMS 密钥权限

如果集中化规则中提供的 KMS 密钥不允许日志使用日志，则集中化规则将无法将日志从源账户传送到目标 CloudWatch 日志组。确保 KMS 密钥策略为 CloudWatch 日志授予必要的权限。有关更多信息，请参阅 步骤 2：设置 KMS 密钥的权限。

客户管理的 KMS 密钥配置

如果您在规则创建期间选择了不集中使用客户管理的 KMS 密钥加密的日志组，则使用客户管理的 KMS 密钥加密的源日志组中的日志事件将被跳过且不会集中化。

目标加密不匹配

如果目标日志组已经存在，其KMS加密配置与集中规则指定的配置不同，并且冲突解决方案设置为SKIP，则记录将被删除并发出`DestinationEncryptionMismatch`错误。例如，当目标采用默认加密但规则指定了客户管理的 KMS 密钥时，就会发生这种情况。

未启用可信访问

必须为 CloudWatch 管理账户和目标账户启用可信访问权限，以提供对日志数据的访问权限。AWS Organizations

来源选择标准

验证您的集中化规则的来源选择标准是否配置正确：

- 账户和区域：确保规则中包含源账户和日志来源区域。来自规则中未指定的账户或区域的日志组将不会被集中化。
- 日志组筛选器：如果您配置了日志组筛选器，则只有符合指定条件的日志组才会被集中化。确认您的日志组选择标准包括您希望集中管理的日志组。
- 组织成员资格：源账户和目标账户必须属于同一个 AWS Organizations 组织。组织以外的账户不能参与集中管理。

已达到日志组配额限制

如果目标账户已达到其日志组配额限制，则无法为集中化创建新的日志组。确认目标账户有足够的配额来容纳来自所有源账户的集中日志组。如果需要，您可以请求增加配额。

已超过日志流名称长度限制

日志流名称有最大长度限制。当集中化将日志流复制到目标账户时，会在日志流名称中添加后缀。如果生成的日志流名称超过允许的最大长度，则记录将被删除，并且会向客户账户发出InvalidLogStream错误。

规则运行状况

在控制台中或使用 GetCentralizationRuleForOrganization API 检查集中化规则的运行状况。如果该规则被标记为 “不健康”，请查看该FailureReason字段以了解有关该问题的具体详细信息。

标签传播健康状态

如果TagPropagationStatus显示Unhealthy，请检查以下TagPropagationFailureReason字段：

- RoleNotAssumable：该服务无法代入目标角色。验证该角色的信任策略是否允许中心化服务相关角色代入该角色，并且该信任策略是否与您的组织 ID sts:ExternalId 相匹配。
- RoleLacksPermissions：角色已代入，但标签 API 调用被拒绝。确保角色的权限策略在目标日志组logs:UntagResource上授予logs:ListTagsForResourcelogs:TagResource、和。

要诊断集中化问题，请在控制台中查看集中化规则的运行状况，检查错误和限制 CloudWatch 指标，并检查 API 调用失败的 AWS CloudTrail 日志。有关集中化指标的更多信息，请参阅集中化指标和维度。

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。

对于需要整合来自多个账户和地区的日志数据的组织，您可以使用 CloudWatch 日志集中将日志组自动复制到中央账户。有关更多信息，请参阅 Cross-account 跨区域日志集中化。

您可以使用本部分中的过程处理日志组和日志流。

在 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 “日志管理”。
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 “日志管理”。
3. 对于 Log Groups（日志组），选择日志组以查看日志流。
4. 在日志组列表中，选择要查看的日志组的名称。
5. 从日志流列表中，选择要查看的日志流的名称。
6. 要更改日志数据的显示方式，请执行下列操作之一：
   - 要展开单个日志事件，请选择该日志事件旁边的箭头。
   - 要展开所有日志事件并以纯文本形式查看它们，请在日志事件列表上方选择 Text。
   - 要筛选日志事件，请在搜索字段中键入所需的搜索筛选条件。有关更多信息，请参阅 使用筛选条件从日志事件创建指标。
   - 要查看指定日期和时间范围的日志数据，请在搜索筛选条件旁边，选择日期和时间旁的箭头。要指定日期和时间范围，请选择 Absolute（绝对）。要选择预定义的分钟数、小时数、天数或周数，请选择 Relative（相对）。还可以在 UTC 和本地时区之间切换。

使用筛选条件模式搜索日志数据

可以使用 指标筛选条件、订阅筛选条件、筛选日志事件和 Live Tail 的筛选条件模式语法 搜索日志数据。您可以搜索日志组中的所有日志流， 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 跨账户可观测性。

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 “日志管理”。
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 “日志管理”。
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。

问题排查

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，选择当前保留期设置，例如 Never Expire。
5. 在保留设置中，对于事件过期时间，选择日志保留值，然后选择保存。

保护日志组不被删除

您可以选择启用删除保护，以防止意外删除重要的日志组。有关删除保护的详细信息，请参阅保护日志组不被删除。

保护日志组不被删除

启用删除保护

您可以在创建新日志组时或在现有日志组上启用删除保护。在创建日志组期间，选择“已启用删除保护”或传递参数`--deletion-protection-enabled`。默认情况下，删除保护未启用。

在现有日志组上启用或禁用删除保护（控制台）

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“日志管理”。
3. 选择要保护的日志组。
4. 选择“操作”、“编辑删除保护”。
5. 在对话框中，查看并提交更改。

如果使用 AWS CLI，要对现有日志组启用删除保护，请执行以下操作：

```
aws logs put-log-group-deletion-protection \
--log-group-identifier "/my-application/logs" \
--deletion-protection-enabled
```

要删除现有日志组的删除保护，请执行以下操作：

```
aws logs put-log-group-deletion-protection \
--log-group-identifier "/my-application/logs" \
--no-deletion-protection-enabled
```

错误处理

如果您尝试删除启用了删除保护的日志组，则会收到一条`ValidationException`消息：“启用了删除保护后无法删除日志组。请先禁用删除保护。”

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 使用具有 256 位高级加密标准 Galois/Counter 模式 (AES-GCM) 的服务器端加密来加密静态日志数据。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。

> **Important**
>
> CloudWatch Log `kms:ViaService` s 现在支持允许日志代表您 AWS KMS 拨打电话。您应该将其添加到在密钥策略或 IA CloudWatch M 中调用 Logs 的角色中。有关更多信息，请参阅 kms: ViaService。

将 KMS 密钥与日志组关联后，该日志组所有新摄取的数据都将使用该密钥加密。这些数据在整个保留期内都以加密格式存储。 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 密钥与现有日志组关联。

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，您可以授予 CloudWatch 日志服务主体和呼叫者角色使用密钥的权限。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。

JSON

```
{
    "Version":"2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::123456789012:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:Describe*"
            ],
            "Resource": "*",
            "Condition": {
```

```
                "ArnEquals": {
                "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-
east-1:111122223333:log-group:log-group-name"
                }
            }
        }
    ]
}
```

本示例中的 Condition 部分对 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。

JSON

```
{
    "Version":"2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::123456789012:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Principal": {
            "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:Describe*"
            ],
            "Resource": "*",
            "Condition": {
```

```
                "ArnLike": {
                "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-
east-1:123456789012:*"
                }
            }
        }
    ]
}
```

步骤 3：为调用方的 IAM 委托人设置权限

向将调用 Logs 的 IAM 委托人（用户或角色）添加 CloudWatch 权限 APIs。所需的权限取决于委托人需要执行的操作。您可以在 AWS KMS 密钥策略中添加这些权限，也可以通过 IAM 对角色本身添加这些权限。 CloudWatch 记录`kms:ViaService`用户 AWS KMS 代表客户拨打电话。有关更多信息，请参阅 kms: ViaService。

将 KMS 密钥与日志组关联的权限

使用`kmsKeyId`参数调用或调`CreateLogGroupAssociateKmsKey`用的 IAM 委托人必须拥有对指定 KMS 密钥的`kms:DescribeKey`权限。如果调用者没有此权限，则 API 调用将失败，并显示为`AccessDeniedException`。

以下示例密钥策略声明授予将 KMS 密钥与日志组关联所需的最低权限：

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::account_id:role/role_name"
  },
  "Action": [
    "kms:Describe*"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": [
        "logs.region.amazonaws.com"
      ]
    }
  }
}
```

读取和写入加密日志数据的权限

如果 IAM 委托人还需要读取或写入加密的日志数据（例如，调用PutLogEventsGetLogEvents、FilterLogEvents、或StartQuery在使用客户托管 KMS 密钥加密的日志组上），则委托人需要额外的 AWS KMS 权限。以下示例密钥策略语句授予关联密钥以及读取或写入加密日志数据所需的全套权限：

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::account_id:role/role_name"
  },
  "Action": [
    "kms:Encrypt",
    "kms:ReEncrypt*",
    "kms:Decrypt",
    "kms:GenerateDataKey*",
    "kms:Describe*"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": [
        "logs.region.amazonaws.com"
      ]
    }
  }
}
```

最佳做法是，将策略范围仅限于将与 AWS KMS 加密日志组交互的角色。

或者，要授予通过 IAM 关联密钥以及读取或写入加密日志数据的全套权限，您可以向调用者角色添加以下策略。您可以将此策略添加到现有角色策略中，也可以将其作为单独的额外策略附加到角色。如果您使用此方法，则作为最佳实践，请将策略范围仅限于将用于日志加密的 AWS KMS 密钥。有关更多信息，请参阅编辑 IAM 策略。

JSON

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

```
        {
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:ReEncrypt*",
                "kms:Decrypt",
                "kms:GenerateDataKey*",
                "kms:Describe*"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "logs.us-east-1.amazonaws.com"
                    ]
                }
            },
            "Resource": "arn:aws:kms:us-east-1:444455556666:key/key_id"
        }
    ]
}
```

最后，使用以下put-key-policy命令添加更新的策略：

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

步骤 4：将 KMS 密钥与日志组关联

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 5：取消密钥与日志组的关联

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 ARNs 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：

JSON

```
{
    "Version":"2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Principal": {
            "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:Describe*"
            ],
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-
east-1:111122223333:log-group:LOG-GROUP-NAME"
                }
            }
        }
```

```
    ]
}
```

5. 输入以下命令以将更新的策略添加到新的 KMS 密钥：

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

6. 输入以下命令以将策略与日志组关联：

```
aws logs associate-kms-key --log-group-name my-log-group --kms-key-id new-key-ARN
```

CloudWatch Logs 现在使用新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

JSON

```
{
    "Version":"2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.region.amazonaws.com"
            },
            "Action": [
```

```
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

9. 输入以下命令以将更新的策略添加到旧的密钥：

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

通过屏蔽帮助保护敏感的日志数据

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**
>
> 标准和不频繁访问日志类中的日志组都支持屏蔽敏感数据。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，检测还取决于是否找到与敏感数据接近的某些关键字。您还可以使用自定义数据标识符来创建针对特定使用场景量身定制的数据标识符。

当检测到与您选择的数据标识符匹配的敏感数据 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
        - 凭据
            - 凭证数据 ARNs 类型的数据标识符
        - 设备标识符
            - 设备数据类型的数据 ARNs 标识符
        - 财务信息
            - 财务数据类型的数据 ARNs 标识符
        - 受保护健康信息（PHI）
            - 受保护 ARNs 的健康信息数据类型 (PHI) 的数据标识符
        - 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) 的数据标识符 ARNs
- 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。
- 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`	*

操作	所需的 IAM 权限	资源
	logs:PutDataProtectionPolicy	*
	logs:CreateLogDelivery	*
	logs:PutResourcePolicy	*
	logs:DescribeResourcePolicies	*
	logs:DescribeLogGroups	*
将 Firehose 作为审计目标来创建数据保护策略	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*
	logs:CreateLogDelivery	*
	firehose:TagDeliveryStream	arn:aws:logs:::deliverystream/ *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	*

操作	所需的 IAM 权限	资源
	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	*
	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* :*

操作	所需的 IAM 权限	资源
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* :* * * * *
将 Firehose 作为审计目标来创建数据保护策略	logs:PutDataProtectionPolicy logs:CreateLogDelivery firehose:TagDeliveryStream	arn:aws:logs:::log-group: *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*

操作	所需的 IAM 权限	资源
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。

JSON

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

```
        {
            "Sid": "AllowDataProtectionAndBucketConfiguration",
            "Effect": "Allow",
            "Action": [
                "logs:GetDataProtectionPolicy",
                "logs:DeleteDataProtectionPolicy",
                "logs:PutDataProtectionPolicy",
                "s3:PutBucketPolicy",
                "firehose:TagDeliveryStream",
                "s3:GetBucketPolicy"
            ],
            "Resource": [
            "arn:aws:firehose:us-east-1:111122223333:deliverystream/delivery-
stream-name",
            "arn:aws:s3:::amzn-s3-demo-destination-bucket",
            "arn:aws:logs:us-east-1:111122223333:log-group:log-group-name:*"
            ]
        }
    ]
}
```

创建账户范围的数据保护策略

您可以使用 CloudWatch 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，选择设置。它位于列表底部附近。
3. 选择日志选项卡。
4. 选择配置。
5. 对于托管数据标识符，选择要为所有日志组审计和屏蔽的数据类型。您可以在选择框中键入以查找所需的标识符。

   建议您仅选择与您的日志数据和业务相关的数据标识符。选择多种类型的数据可能会导致误报。

   有关您可以保护的数据类型的详细信息，请参阅 您可以保护的数据类型。
6. （可选）如果您想使用自定义数据标识符审核和屏蔽其他类型的数据，请选择添加自定义数据标识符。然后输入数据类型的名称和用于在日志事件中搜索该类型数据的正则表达式。有关更多信息，请参阅 自定义数据标识符。

   单个数据保护策略最多可包含 10 个自定义数据标识符。定义自定义数据标识符的每个正则表达式不得超过 200 个字符。
7. （可选）选择要将审核结果发送到的一项或多项服务。即使您选择不将审计结果发送到任何这些服务，您选择的敏感数据类型也仍将屏蔽。
8. 选择 Activate data protection（激活数据保护）。

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。如果指定日志组、Amazon Data Firehose 流和 S3 存储桶等目标，则它们必须已经存在。有关审计结果报告的示例，请参阅 [审计结果报告]。
- 第二个块必须包含 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。

下面是仅使用托管数据标识符的数据保护策略示例。此策略屏蔽电子邮件地址和美国驾驶执照。

有关指定自定义数据标识符的策略信息，请参阅 [在数据保护策略中使用自定义数据标识符]。

```
{
    "Name": "data-protection-policy",
    "Description": "test description",
    "Version": "2021-06-01",
    "Statement": [{
            "Sid": "audit-policy",
            "DataIdentifier": [
                "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
                "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
            ],
            "Operation": {
                "Audit": {
                    "FindingsDestination": {
                        "CloudWatchLogs": {
                            "LogGroup": "EXISTING_LOG_GROUP_IN_YOUR_ACCOUNT,"
                        },
                        "Firehose": {
                            "DeliveryStream": "EXISTING_STREAM_IN_YOUR_ACCOUNT"
                        },
```

```
                        "S3": {
                            "Bucket": "EXISTING_BUCKET"
                        }
                    }
                }
            }
        },
        {
            "Sid": "redact-policy",
            "DataIdentifier": [
                "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
                "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
            ],
            "Operation": {
                "Deidentify": {
                    "MaskConfig": {}
                }
            }
        }
    ]
}
```

为单个日志组创建数据保护策略

您可以使用 CloudWatch 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. 对于托管数据标识符，选择要在此日志组中审计和屏蔽的数据类型。您可以在选择框中键入以查找所需的标识符。

   建议您仅选择与您的日志数据和业务相关的数据标识符。选择多种类型的数据可能会导致误报。

   有关可以使用托管数据标识符保护哪些类型的数据的详细信息，请参阅 您可以保护的数据类型。
6. （可选）如果您想使用自定义数据标识符审核和屏蔽其他类型的数据，请选择添加自定义数据标识符。然后输入数据类型的名称和用于在日志事件中搜索该类型数据的正则表达式。有关更多信息，请参阅 自定义数据标识符。

   单个数据保护策略最多可包含 10 个自定义数据标识符。定义自定义数据标识符的每个正则表达式不得超过 200 个字符。
7. （可选）选择要将审核结果发送到的一项或多项服务。即使您选择不将审计结果发送到任何这些服务，您选择的敏感数据类型也仍将屏蔽。
8. 选择 Activate data protection（激活数据保护）。

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。如果指定日志组、Amazon Data Firehose 流和 S3 存储桶等目标，则它们必须已经存在。有关审计结果报告的示例，请参阅 审计结果报告。

- 第二个块必须包含 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）。
- 使用包含取消屏蔽 (@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 日志见解语言查询语法。
- 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特定使用场景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
  - 凭据
    - 凭证数据 ARNs 类型的数据标识符
  - 设备标识符
    - 设备数据类型的数据 ARNs 标识符
  - 财务信息
    - 财务数据类型的数据 ARNs 标识符
  - 受保护健康信息（PHI）
    - 受保护 ARNs 的健康信息数据类型 (PHI) 的数据标识符
  - 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) 的数据标识符 ARNs

- 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

本节包含有关可使用托管数据标识符保护的数据类型的信息，以及哪些国家和地区与每种类型的数据相关。

对于某些类型的敏感数据， CloudWatch Logs 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）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`	凭据
`CreditCardExpiration`	财务
`CreditCardNumber`	财务
`CreditCardSecurityCode`	财务
`EmailAddress`	个人
`IpAddress`	个人

标识符	类别
LatLong	个人
Name	个人
OpenSshPrivateKey	凭据
PgpPrivateKey	凭据
PkcsPrivateKey	凭据
PuttyPrivateKey	凭据
VehicleIdentificationNumber	个人

区域相关数据标识符必须包含标识符名称，然后是连字符，最后是两个字母（ISO 3166-1 alpha-2）代码。例如 DriversLicense-US。

支持的标识符，必须包含两个字母的国家或地区代码

标识符	类别	国家/地区和语言
BankAccountNumber	财务	DE、ES、FR、GB、IT、US
CepCode	个人	BR
Cnpj	个人	BR
CpfCode	个人	BR
DriversLicense	个人	AT、AU、BE、 BG、CA、CY、 CZ、DE、DK、EE、ES、FI、 FR、GB、GR、 HR、HU、IE、IT、LT、LU、 LV、MT、NL、 PL、PT、RO、SE、SI、SK、 US

标识符	类别	国家/地区和语言
DrugEnforcementAgencyNumber	健康	美国
ElectoralRollNumber	个人	GB
HealthInsuranceCardNumber	健康	EU
HealthInsuranceClaimNumber	健康	美国
HealthInsuranceNumber	健康	FR
HealthcareProcedureCode	健康	美国
IndividualTaxIdentificationNumber	个人	美国
InseeCode	个人	FR
MedicareBeneficiaryNumber	健康	美国
NationalDrugCode	健康	美国
NationalIdentificationNumber	个人	DE、ES、IT
NationalInsuranceNumber	个人	GB
NationalProviderId	健康	美国
NhsNumber	健康	GB
NieNumber	个人	ES
NifNumber	个人	ES
PassportNumber	个人	CA、DE、ES、FR、GB、IT、US
PermanentResidenceNumber	个人	CA
PersonalHealthNumber	健康	CA

标识符	类别	国家/地区和语言
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

凭据

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

数据类型	数据标识符 ID	所需关键字	国家和地区
PuTTY 私有密钥	`PuttyPrivateKey`	无	全部

凭证数据 ARNs 类型的数据标识符

以下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 (ARNs)。

凭证数据标识符 ARNs
`arn:aws:dataprotection::aws:data-identifier/AwsSecretKey`
`arn:aws:dataprotection::aws:data-identifier/OpenSshPrivateKey`
`arn:aws:dataprotection::aws:data-identifier/PgpPrivateKey`
`arn:aws:dataprotection::aws:data-identifier/PkcsPrivateKey`
`arn:aws:dataprotection::aws:data-identifier/PuttyPrivateKey`

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 ARNs 标识符

以下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 (ARNs)。

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`	可以。不同的关键字适用于不同的国家/地区。有关详细信息，请参阅本节后面的 Keywords for bank account numbers（银行账号的关键字）表。	法国、德国、意大利、西班牙、英国、美国	包括由最多 34 个字母数字字符组成的国际银行账号 (IBANs)，包括国家/地区代码等元素。
信用卡到期日期	`CreditCardExpiration`	`exp d`, `exp m`, `exp y`, `expiration` , `expiry`	全部	
信用卡号	`CreditCardNumber`	`account number`, `american express`, `amex`, `bank card`, `card`,	全部	检测要求数据为符合

数据类型	数据标识符 ID	所需关键字	国家和地区	注意
		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		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

使用以下关键字来获取银行账号。这包括由最多 34 个字母数字字符组成的国际银行账号 (IBANs)，包括国家/地区代码等元素。

国家/地区	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,

国家/地区	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 ARNs 标识符

以下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 (ARNs)。

财务数据标识符 ARNs
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-DE
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-ES
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-FR
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-GB
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-IT
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-US
arn:aws:dataprotection::aws:data-identifier/CreditCardExpiration
arn:aws:dataprotection::aws:data-identifier/CreditCardNumber
arn:aws:dataprotection::aws:data-identifier/CreditCardSecurityCode

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）	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）代码	HealthcareProcedureCode	current procedural terminology , hcpcs, healthcare common procedure coding system	美国
医疗保险受益人号码（MBN）	MedicareBeneficiaryNumber	mbi, medicare beneficiary	美国
国家药品编码（NDC）	NationalDrugCode	national drug code, ndc	美国
国家提供商识别码（NPI）	NationalProviderId	hipaa, n.p.i., national provider, npi	美国

数据类型	数据标识符 ID	所需关键字	国家和地区
国家健康服务（NHS, National Health Service）号码	NhsNumber	national health service, NHS	英国
个人健康号码	PersonalHealthNumber	canada healthcare number, msp number, care number, phn, soins de santé	加拿大

受保护 ARNs 的健康信息数据类型 (PHI) 的数据标识符

下面列出了可在受保护的健康信息 (PHIARNs) 数据保护策略中使用的数据标识符 Amazon 资源名称 ()。

PHI 数据标识符 ARNs
arn:aws:dataprotection::aws:data-identifier/DrugEnforcementAgencyNumber-US
arn:aws:dataprotection::aws:data-identifier/HealthcareProcedureCode-US
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceCardNumber-EU
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceClaimNumber-US
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceNumber-FR
arn:aws:dataprotection::aws:data-identifier/MedicareBeneficiaryNumber-US
arn:aws:dataprotection::aws:data-identifier/NationalDrugCode-US

PHI 数据标识符 ARNs
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	Any	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	可以。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	可以。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）	`InseeCode`	可以。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`	可以。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）	`NationalInsuranceNumber`	`insurance no.`, `insurance number`, `insurance#` , `national insurance number`, `nationalinsurance#` , `nationali nsurancenumber` , `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`	可以。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`	可以。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`	可以。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
Phone number（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	可以。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）	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	可以。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	Any	

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）	LatLong	coordinate ，coordinates ，latlong， latitude longitude ，location，position	Any	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）格式的坐标，例如 41°56.9168'N 87°39.3187'W，也不支持度分秒

数据类型	数据标识符 ID	所需关键字	国家和地区	注意
				（DMS）格式的坐标，例如 41°56'55.0104"N 87°39'19.1196"W。
全名	Name	无	Any	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	Any	CloudWatch 日志 VINs 可以检测包含 17 个字符的序列并符合 ISO 3779 和 3780 标准的日志。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）号码（巴西）。

下表列出了 CloudWatch Logs 识别的特定国家和地区的关键字。

国家或地区	关键词
巴西	registro geral, rg
法国	assurance sociale, carte nationale d’identit é, cni, code sécurité sociale, French social security number, fssn#, insee, insurance number, national id number, nationalid#, numéro d'assurance, sécurité sociale, sécurité

国家或地区	关键词
	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) 的数据标识符 ARNs

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 ()。ARNs

PII 数据标识符 ARNs
`arn:aws:dataprotection::aws:data-identifier/Address`
`arn:aws:dataprotection::aws:data-identifier/CepCode-BR`
`arn:aws:dataprotection::aws:data-identifier/Cnpj-BR`
`arn:aws:dataprotection::aws:data-identifier/CpfCode-BR`
`arn:aws:dataprotection::aws:data-identifier/DriversLicense-AT`

PII 数据标识符 ARNs
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 ARNs
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 ARNs
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 ARNs
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 (CDIs) 允许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。 IDs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数据保护策略中指定自定义数据标识符。与托管数据标识符类似，账户级策略中定义的自定义数据标识符与日志组级策略中定义的自定义数据标识符结合使用。

在控制台中使用自定义数据标识符

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** 作为名称，并输入 **EmployeeID-\d{9}** 作为正则表达式。此正则表达式将检测并屏蔽 EmployeeID- 后面带有 9 个数字的日志事件。例如，EmployeeID-123456789

在数据保护策略中使用自定义数据标识符

如果您使用 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涉及公司特定员工的日志事件。 IDs

1. 在您的数据保护策略中创建一个 Configuration 块。
2. 为您的自定义数据标识符输入 Name。例如 **EmployeeId**。
3. 为您的自定义数据标识符输入 Regex。例如 **EmployeeID-\d{9}**。此正则表达式将匹配包含 EmployeeID- 且 EmployeeID- 后面有 9 位数字的日志事件。例如，EmployeeID-123456789
4. 请参阅策略声明中的以下自定义数据标识符。

```
{
    "Name": "example_data_protection_policy",
    "Description": "Example data protection policy with custom data identifiers",
    "Version": "2021-06-01",
    "Configuration": {
      "CustomDataIdentifier": [
        {"Name": "EmployeeId", "Regex": "EmployeeId-\\d{9}"}
      ]
    },
    "Statement": [
        {
            "Sid": "audit-policy",
            "DataIdentifier": [
                "EmployeeId"
            ],
            "Operation": {
                "Audit": {
                    "FindingsDestination": {
                        "S3": {
                            "Bucket": "EXISTING_BUCKET"
                        }
                    }
                }
            }
        },
        {
            "Sid": "redact-policy",
            "DataIdentifier": [
            "EmployeeId"
            ],
            "Operation": {
                "Deidentify": {
                    "MaskConfig": {
                    }
                }
            }
        }
    ]
}
```

5. （可选）根据需要继续向 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。

在摄取期间转换日志

通过日志转换和充实，在将所有日志摄入日志时，您可以将所有日志标准化为一致且上下文丰富的格式。 CloudWatch 您可以为常见 AWS 服务（例如 AWS WAF 和 Amazon Route 53）使用预先配置的模板为日志添加结构，也可以使用 Grok 等原生解析器构建自定义转换器。您还可以重命名现有属性，并向日志添加其他元数据（例如，账户 ID 和区域）。

日志转换有助于简化和缩短应用程序中的日志查询，并有助于简化在日志上创建警报的过程。此功能通过主要 AWS 日志源（如 VPC 流日志、Route 53 和）的 out-of-the-box转换模板为常见日志类型提供转换 Amazon RDS for PostgreSQL。您可以使用预配置的转换模板，也可以创建自定义转换器以满足您的需求。

日志转换可帮助您管理从格式和属性名称差异很大的源发出的日志。

创建转换器后，摄取的日志事件将以标准格式转换和存储。您可以利用这些转换后的日志，通过以下功能加速您的分析体验：

- 字段索引
- CloudWatch 日志见解已发现字段
- 使用指标筛选条件灵活地设置警报
- 通过订阅筛选条件进行转发
- 使用 Contributor Insights 从日志事件创建指标数据，您可以选择让 Contributor Insights 规则在转换日志事件之前或之后对其进行评估。

转换仅在日志摄取期间进行。您无法转换已摄取的日志事件。转换操作不可逆。原始日志和转换后的日志都存储在具有相同保留策略的 CloudWatch 日志中。日志转换和丰富功能包含在现有的标准日志类摄取价格中。日志存储成本将根据转换后的日志大小计算，这可能会超过原始日志量。

> **Important**
>
> 转换日志事件后，您必须使用 Lo CloudWatch gs Insights 查询来查看转换后的日志版本。GetLogEvents和 FilterLogEvents操作仅返回日志事件在转换之前的原始版本。

> **Important**
>
> 尽管单个日志事件最多 PutLogEvents 允许 1MB，但日志转换只能处理大小小于 512kb 的日志事件。任何大于 512kb 的日志事件都将在转换中失败并发出错误。的总大小仍然 PutLogEvents 可以超过 512kb。

除了转换为不同的格式外，您还可以使用其他上下文（例如，账户 ID、区域和关键字）丰富日志。这些信息是从日志组名称和静态关键字中提取的。

日志转换可帮助您处理从格式和属性名称差异很大的源发出的日志。

仅标准日志类中的日志组支持日志转换和丰富功能。

您可以为单个日志组创建转换器，也可以创建适用于账户中所有或多个日志组的账户级转换器。如果某个日志组具有日志组级转换器，则该转换器会覆盖原本适用于该日志组的任何账户级转换器。

主题

- 创建和管理日志转换器
- 可配置的解析器类型处理器
- 用于 AWS 出售日志的内置处理器
- 字符串变异处理器
- JSON 变异处理器
- 数据类型转换器处理器
- 转换指标和错误

创建和管理日志转换器

日志转换器包括一个或多个一起处于逻辑管线中的处理器。每个处理器都按照它们在转换器配置中列出的顺序依次应用于日志事件。

有些处理器属于解析器类型。每个转换器必须至少有一个解析器，并且转换器中的第一个处理器必须是解析器。

有些解析器是针对特定类型的 AWS 公开发布的日志进行了配置的内置解析器。

其他处理器类型包括字符串变异器、JSON 变异器和数据处理器。

您可以为单个日志组创建转换器，也可以创建适用于账户中所有或多个日志组的账户级转换器。如果某个日志组具有日志组级转换器，则该转换器会覆盖原本适用于该日志组的任何账户级转换器。在您的账户中，一个区域最多可以有 20 个账户级转换器。

在创建转换器时，必须遵循以下准则：

- 如果您为某种类型的已 AWS 售日志包含预配置的解析器，则该解析器必须是转换器中列出的第一个处理器。只能在转换器中包含一个此类处理器。
- 只能在转换器中包含一个 `grok` 处理器。
- 转换器必须至少包含一个解析器类型的处理器。最多可以包含五个解析器类型的处理器。此五个限制包括内置解析器和可配置的解析器。
- 转换器最多可包含 20 个处理器。
- 只能在转换器中包含一个 addKeys 处理器。
- 只能在转换器中包含一个 copyValue 处理器。
- 每个转换器最多可以从日志事件中提取 200 个字段。
- 每个日志事件必须低于 512KB。日志事件的总大小仍然可以超过 512KB。

主题

- 创建账户级转换器策略
- 编辑或删除账户级转换器策略
- 从头开始创建对 log-group-level数转换器
- 通过复制现有的 log-group-level变压器来创建变压器
- 编辑变 log-group-level压器
- 删除变 log-group-level压器

创建账户级转换器策略

请按照本节中的步骤创建适用于账户中的所有日志组或者日志组名称以相同字符串（前缀）开头的多个日志组的转换器策略。一个区域中最多可以有 20 个账户级转换器策略。

您不能在同一区域创建两个使用相同前缀或一个前缀包含在另一个前缀中的转换器策略。例如，如果您为字符串前缀 `/aws/lambda` 创建了一个转换器策略，则不能再使用前缀 `/aws` 创建另一个转换器策略。但是，您可以为 `/aws/lambda` 创建一个转换器策略，而为 `/aws/waf` 创建另一个转换器策略

创建账户级转换器策略

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在左侧导航窗格上，选择设置，然后选择日志选项卡。
3. 在账户的转换器策略部分，选择创建转换器策略。
4. 对于转换器策略名称，输入新策略的名称。
5. 对于选择日志组，请执行下列操作之一：
   - 选择所有标准日志组，将转换器策略应用于账户中的所有标准类日志组。
   - 选择按前缀匹配选择日志组，将策略应用于名称以相同字符串开头的日志组子集。然后，在选择标准中输入这些日志组的前缀。
6. 在选择解析器区域中，使用解析器选择要包含在转换器中的解析器。

   如果它是预先配置的解析器，则无需为其指定任何配置。 AWS

   如果是其他解析器，则需要指定其配置。有关更多信息，请参阅可配置的解析器类型处理器中该处理器的相关信息。
7. 要添加其他处理器，请选择选择处理器。然后在处理器框中选择所需的处理器，并填写配置参数。

   请记住，处理器会按照您将日志事件添加到转换器的顺序对其进行操作。
8. （可选）要添加其他处理器，请选择 + 处理器并重复上一步。
9. （可选）您可以随时测试迄今为止您根据示例日志事件构建的转换器。为此，请在转换器预览部分执行以下操作之一：
   - 在选择日志组中最多选择五个日志组，然后选择加载最新的日志事件。然后选择测试转换器。
   - 将日志事件直接复制到示例日志事件中，然后选择测试转换器。

   随后会显示日志的转换后版本。
10. 添加完处理器并对示例日志的测试感到满意后，请选择保存。
11. 完成后，选择创建。

编辑或删除账户级转换器策略

请按照本部分中的步骤编辑或删除账户级转换器策略。

编辑或删除账户级转换器策略

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

2. 在左侧导航窗格上，选择设置，然后选择日志选项卡。
3. 在转换器账户策略部分中，选择管理。
4. 选择要管理的转换器策略旁边的按钮，然后选择编辑或删除。

   如果您正在编辑策略，请参阅可配置的解析器类型处理器中的步骤 5-11 以查看您的选项。

从头开始创建对 log-group-level数转换器

使用这些步骤从头开始创建 log-group-level变压器。

使用控制台为日志组创建日志转换器

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. 选择要为其创建转换器的日志组。
4. 选择转换器选项卡。您可能需要向右滚动选项卡列表才能看到它。
5. 选择创建转换器。
6. 在选择解析器框中，选择要包含在转换器中的解析器。

   如果它是预先配置的解析器，则无需为其指定任何配置。 AWS

   如果是其他解析器，则需要指定其配置。有关更多信息，请参阅可配置的解析器类型处理器中该处理器的相关信息。
7. 要添加其他处理器，请选择 + 添加处理器。然后在选择处理器框中选择所需的处理器，并填写配置参数。

   请记住，处理器会按照您将日志事件添加到转换器的顺序对其进行操作。
8. （可选）您可以随时测试迄今为止您根据示例日志事件构建的转换器。为此，请执行以下操作：

   - 在转换预览部分，选择加载示例日志以从此转换器适用的日志组加载示例日志事件，或者将日志事件粘贴到文本框中。

     选择测试转换器。随后会显示日志的转换后版本
9. 添加完处理器并对示例日志的测试感到满意后，请选择保存。

使用从头 AWS CLI 开始创建对数转换器

- 使用 `aws logs put-transformer` 命令。当使用 `parseJSON` 作为第一个处理器时，您必须使用 `@message` 作为源字段来解析整个日志事件。在初始 JSON 解析后，您可以在后续处理器中操作特定字段。以下示例创建了一个包含 `parseJSON` 和 `addKeys` 处理器的转换器：

```
 aws logs put-transformer \
  --transformer-config '[{"parseJSON":{"source":"@message"}},{"addKeys":
{"entries":[{"key":"metadata.transformed_in","value":"CloudWatchLogs"},
{"key":"feature","value":"Transformation"}]}},{"trimString":{"withKeys":
["status"]}}]' \
  --log-group-identifier my-log-group-name
```

通过复制现有的 log-group-level变压器来创建变压器

您可以使用控制台复制现有转换器的 JSON 配置。然后，您可以使用该代码创建相同的转换器 AWS CLI，也可以先修改配置。

通过复制现有日志转换器来创建日志转换器

1. 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. 选择包含要复制的转换器的日志组。
4. 选择转换选项卡。您可能需要向右滚动选项卡列表才能看到它。
5. 选择管理转换器。
6. 选择复制转换器。此时会将转换器 JSON 复制到剪贴板。
7. 创建一个文件并粘贴转换器配置。在此示例中，我们将调用文件 `CopiedTransformer.json`
8. 使用创建具有 AWS CLI 该配置的新变压器。

```
aws logs put-transformer --log-group-identifier my-log-group-name \
--transformer-config file://CopiedTransformer.json
```

编辑变 log-group-level压器

请按照以下步骤编辑现有的日志转换器。

编辑日志转换器

1. 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. 选择包含要编辑的转换器的日志组。
4. 选择转换选项卡。您可能需要向右滚动选项卡列表才能看到它。
5. 选择管理转换器。
6. 在解析器和处理器部分，进行更改。
7. 要添加其他处理器，请选择 + 添加处理器。然后在处理器框中选择所需的处理器，并填写配置参数。

   请记住，处理器会按照您将日志事件添加到转换器的顺序对其进行操作。
8. （可选）您可以随时测试迄今为止您根据示例日志事件构建的转换器。为此，请执行以下操作：

   - 在转换预览部分，选择加载示例日志以从此转换器适用的日志组加载示例日志事件，或者将日志事件粘贴到文本框中。

     选择测试转换。随后会显示日志的转换后版本
9. 添加完处理器并对示例日志的测试感到满意后，请选择保存。

删除变 log-group-level压器

请按照以下步骤删除日志转换器。

删除日志转换器

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. 选择包含要编辑的转换器的日志组。
4. 选择转换选项卡。您可能需要向右滚动选项卡列表才能看到它。
5. 选择删除。
6. 在确认框中，选择删除策略。

可配置的解析器类型处理器

本节包含有关可在日志事件转换器中使用的可配置数据解析器处理器的信息。

目录

- parseJSON
- grok
  - Grok 示例
    - 示例 1：使用 grok 从非结构化日志中提取字段
    - 示例 2：搭配使用 grok 和 parseJSON 从 JSON 日志事件中提取字段
    - 示例 3：FIELD_NAME 中带有点注释的 Grok 模式
  - 支持的 Grok 模式
    - 常见日志格式示例
      - Apache 日志示例
      - NGINX 日志示例
      - Syslog 协议（RFC 5424）日志示例
- csv
- parseKeyValue

parseJSON

parseJSON 处理器解析 JSON 日志事件，并将提取的 JSON 键值对插入到目标位置。如果没有指定目标位置，则处理器会将键值对放置在根节点下。当使用 `parseJSON` 作为第一个处理器时，您必须使用 `@message` 作为源字段来解析整个日志事件。在初始 JSON 解析后，您可以在后续处理器中操作特定字段。

原始 `@message` 内容不变，新键会添加到消息中。

字段	说明	必填？	默认	限制
source	日志事件中待解析字段的路径。请使用点表示法访问子字段。例如，`store.book`	否	`@message`	最大长度：128 最大嵌套键深度：3

字段	说明	必填？	默认	限制
destination	已解析 JSON 的目标字段	否	Parent JSON node	最大长度：128 最大嵌套键深度：3

示例

假设摄取的日志事件如下所示：

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

如果我们有此 parseJSON 处理器：

```
[
   {
        "parseJSON": {
            "destination": "new_key"
        }
   }
]
```

转换后的日志事件将如下所示。

```
{
    "new_key": {
        "outer_key": {
            "inner_key": "inner_value"
        }
    }
}
```

grok

使用 grok 处理器通过模式匹配来解析和构造非结构化数据。此处理器还可以从日志消息中提取字段。

字段	说明	必填？	默认	限制	注意
source	要应用 Grok 匹配的字段路径	否	@messag	最大长度：128 最大嵌套键深度：3	
match	要与日志事件匹配的 grok 模式	是		最大长度：512 最大 grok 模式：20 某些 grok 模式类型有单独的使用限制。以下模式可以任意组合使用，最多可使用五次：{URI, URIPARAM, URIPATHPARAM, SPACE, DATA, GREEDYDATA, GREEDYDATA_MULTILINE} Grok 模式不支持类型转换。 对于常见的日志格式模式（APACHE_ACCESS_LOG、NGINX_ACCESS_LOG、SYSLOG5424），仅支持在常见日志模式之后包含 DATA、GREEDYDATA 或 GREEDYDATA_MULTILINE 模式。	查看所有支持的 Grok 模式

Grok 模式的结构

下面是支持的 grok 模式结构：

```
%{PATTERN_NAME:FIELD_NAME}
```

- PATTERN_NAME：指用于匹配特定类型数据的预定义正则表达式。仅支持预定义的 grok 模式。不允许创建自定义模式。
- FIELD_NAME：为提取的值分配名称。FIELD_NAME 是可选的，但是如果您不指定此值，则提取的数据将从转换后的日志事件中删除。如果 FIELD_NAME 使用点表示法（例如“parent.child”），则将其视为 JSON 路径。
- 类型转换：不支持显式类型转换。使用 p TypeConverter ro cessor 转换由 grok 提取的任何值的数据类型。

要创建更复杂的匹配表达式，您可以组合多个 grok 模式。最多可以组合 20 个 grok 模式来匹配一个日志事件。例如，可以使用此模式组合 `%{NUMBER:timestamp} [%{NUMBER:db} %{IP:client_ip}:%{NUMBER:client_port}] %{GREEDYDATA:data}` 从 Redis 慢日志条目中提取字段，如下所示：

```
1629860738.123456 [0 127.0.0.1:6379] "SET" "key1" "value1"
```

Grok 示例

示例 1：使用 grok 从非结构化日志中提取字段

日志示例：

```
293750 server-01.internal-network.local OK "[Thread-000] token generated"
```

使用的转换器：

```
[
     {
         "grok": {
             "match": "%{NUMBER:version} %{HOSTNAME:hostname} %{NOTSPACE:status}
 %{QUOTEDSTRING:logMsg}"
         }
    }
]
```

输出：

```
{
  "version": "293750",
  "hostname": "server-01.internal-network.local",
  "status": "OK",
  "logMsg": "[Thread-000] token generated"
}
```

日志示例：

```
23/Nov/2024:10:25:15 -0900 172.16.0.1 200
```

使用的转换器：

```
[
    {
        "grok": {
            "match": "%{HTTPDATE:timestamp} %{IPORHOST:clientip}
 %{NUMBER:response_status}"
        }
    }
]
```

输出：

```
{
  "timestamp": "23/Nov/2024:10:25:15 -0900",
  "clientip": "172.16.0.1",
  "response_status": "200"
}
```

示例 2：搭配使用 grok 和 parseJSON 从 JSON 日志事件中提取字段

日志示例：

```
{
    "timestamp": "2024-11-23T16:03:12Z",
    "level": "ERROR",
    "logMsg": "GET /page.html HTTP/1.1"
}
```

使用的转换器：

```
[
     {
        "parseJSON": {}
    },
    {
         "grok": {
            "source": "logMsg",
             "match": "%{WORD:http_method} %{NOTSPACE:request} HTTP/
%{NUMBER:http_version}"
         }
    }
]
```

输出：

```
{
  "timestamp": "2024-11-23T16:03:12Z",
  "level": "ERROR",
  "logMsg": "GET /page.html HTTP/1.1",
  "http_method": "GET",
  "request": "/page.html",
  "http_version": "1.1"
}
```

示例 3：FIELD_NAME 中带有点注释的 Grok 模式

日志示例：

```
192.168.1.1 GET /index.html?param=value 200 1234
```

使用的转换器：

```
[
    {
        "grok": {
            "match": "%{IP:client.ip} %{WORD:method} %{URIPATHPARAM:request.uri}
 %{NUMBER:response.status} %{NUMBER:response.bytes}"
        }
    }
]
```

输出：

```
{
  "client": {
    "ip": "192.168.1.1"
  },
  "method": "GET",
  "request": {
    "uri": "/index.html?param=value"
  },
  "response": {
    "status": "200",
    "bytes": "1234"
  }
}
```

支持的 Grok 模式

下表列出了 grok 处理器支持的模式。

常规 grok 模式

Grok 模式	说明	最大模式限制	示例
USERNAME 或 USER	匹配一个或多个字符，可以包括小写字母（a-z）、大写字母（A-Z）、数字（0-9）、点（.）、下划线（_）或连字符（-）	20	输入：`user123.name-TEST` 模式：`%{USERNAME:name}` 输出：`{"name": "user123.name-TEST"}`
INT	匹配可选的加号或减号，后跟一个或多个数字。	20	输入：`-456` 模式：`%{INT:num}` 输出：`{"num": "-456"}`
BASE10NUM	匹配带有可选符号和小数点的整数或浮点数	20	输入：`-0.67` 模式：`%{BASE10NUM:num}`

Grok 模式	说明	最大模式限制	示例
			输出：{"num": "-0.67"}
BASE16NUM	匹配带有可选符号（+ 或 -）和可选 0x 前缀的十进制和十六进制数字	20	输入：+0xA1B2 模式：%{BASE16NUM:num} 输出：{"num": "+0xA1B2"}
POSINT	匹配不带前导零的正整数，由一个或多个数字组成（1-9 后跟 0-9）	20	输入：123 模式：%{POSINT:num} 输出：{"num": "123"}
NONNEGINT	匹配任何整数（由一个或多个数字 0-9 组成），包括零和带前导零的数字。	20	输入：007 模式：%{NONNEGINT:num} 输出：{"num": "007"}
WORD	匹配由一个或多个单词字符（\w）组成的完整单词，包括字母、数字和下划线	20	输入：user_123 模式：%{WORD:user} 输出：{"user": "user_123"}
NOTSPACE	匹配一个或多个非空格字符。	5	输入：hello_world123 模式：%{NOTSPACE:msg} 输出：{"msg": "hello_world123"}
SPACE	匹配零个或多个空格字符。	5	输入：" " 模式：%{SPACE:extra} 输出：{"extra": " "}

Grok 模式	说明	最大模式限制	示例
DATA	匹配任意字符（换行符除外）零次或多次，不贪婪。	5	输入：`abc def ghi` 模式：`%{DATA:x} %{DATA:y}` 输出：`{"x": "abc", "y": "def ghi"}`
GREEDYDATA	匹配任意字符（换行符除外）零次或多次，贪婪。	5	输入：`abc def ghi` 模式：`%{GREEDYDATA:x} %{GREEDYDATA:y}` 输出：`{"x": "abc def", "y": "ghi"}`

GREEDYDATA_MULTILINE	匹配任意字符（包括换行符）零次或多次，贪婪。	1	输入：
			
			`abc`
			
			`def`
			
			`ghi`
			
			模式：`%{GREEDYDATA_MULTILINE:data}`
			
			输出：`{"data": "abc\ndef\nghi"}`

QUOTEDSTRING	将带引号的字符串（单引号或双引号）与转义字符匹配。	20	输入：`"Hello, world!"` 模式：`%{QUOTEDSTRING:msg}` 输出：`{"msg": "Hello, world!"}`

Grok 模式	说明	最大模式限制	示例
UUID	匹配标准 UUID 格式：8 个十六进制字符，后跟三组 4 个十六进制字符，以 12 个十六进制字符结尾，全部用连字符分隔。	20	输入：550e8400-e29b-41d4-a716-446655440000 模式：%{UUID:id} 输出：{"id": "550e8400-e29b-41d4-a716-446655440000"}
URN	匹配 URN（统一资源名称）语法。	20	输入：urn:isbn:0451450523 模式：%{URN:urn} 输出：{"urn": "urn:isbn:0451450523"}

AWS Grok 图案

模式	说明	最大模式限制	示例
进行筛选	匹配 AWS Amazon 资源名称 (ARNs)，捕获分区（awsaws-cn、或aws-us-gov ）、服务、区域、账户 ID 以及最多 5 个由斜杠分隔的分层资源标识符。它与冒号之间缺少 ARNs 的信息不匹配。	5	输入：arn:aws:iam:us-east-1:123456789012:user/johndoe 模式：%{ARN:arn} 输出：{"arn": "arn:aws:iam:us-east-1:123456789012:user/johndoe"}

联网 grok 模式

Grok 模式	说明	最大模式限制	示例
CISCOMAC	匹配 4-4-4 十六进制格式的 MAC 地址。	20	输入：0123.4567.89AB 模式：%{CISCOMAC:MacAddress} 输出：{"MacAddress": "0123.4567.89AB"}
WINDOWSMAC	将十六进制格式的 MAC 地址与连字符匹配	20	输入：01-23-45-67-89-AB 模式：%{WINDOWSMAC:MacAddress} 输出：{"MacAddress": "01-23-45-67-89-AB"}
COMMONMAC	将十六进制格式的 MAC 地址与冒号匹配。	20	输入：01:23:45:67:89:AB 模式：%{COMMONMAC:MacAddress} 输出：{"MacAddress": "01:23:45:67:89:AB"}
MAC	匹配 CISCOMAC、WINDOWSMAC 或 COMMONMAC grok 模式之一	20	输入：01:23:45:67:89:AB 模式：%{MAC:m1} 输出：{"m1":"01:23:45:67:89:AB"}
IPV6	匹配 IPv6 地址，包括压缩表单和 IPv4映射 IPv6 地址。	5	输入：2001:db8:3333:4444:5555:6666:7777:8888

Grok 模式	说明	最大模式限制	示例
			模式：%{IPV6:ip} 输出：{"ip": "2001:db8:3333:4444:5555:6666:7777:8888"}
IPV4	匹配 IPv4 地址。	20	输入：192.168.0.1 模式：%{IPV4:ip} 输出：{"ip": "192.168.0.1"}
IP	匹配% {IPv6} 支持 IPv4 的地址或% {IPv4} 支持的地址 IPv6	5	输入：192.168.0.1 模式：%{IP:ip} 输出：{"ip": "192.168.0.1"}
HOSTNAME 或 HOST	匹配域名，包括子域	5	输入：server-01.internal-network.local 模式：%{HOST:host} 输出：{"host": "server-01.internal-network.local"}

Grok 模式	说明	最大模式限制	示例
IPORHOST	匹配主机名或 IP 地址	5	输入：2001:db8:3333:4444:5555:6666:7777:8888 模式：%{IPORHOST:ip} 输出：{"ip": "2001:db8:3333:4444:5555:6666:7777:8888"}
HOSTPORT	匹配 %{IPORHOST} 模式支持的 IP 地址或主机名，后跟冒号和端口号，并将端口号捕获为输出中的“PORT”。	5	输入：192.168.1.1:8080 模式：%{HOSTPORT:ip} 输出：{"ip":"192.168.1.1:8080","PORT":"8080"}
URIHOST	匹配 %{IPORHOST} 模式支持的 IP 地址或主机名，可后跟冒号和端口号，并将端口捕获为“port”（如果存在）。	5	输入：example.com:44310.0.0.1 模式：%{URIHOST:host} %{URIHOST:ip} 输出：{"host":"example.com:443","port":"443","ip":"10.0.0.1"}

Path grok 模式

Grok 模式	说明	最大模式限制	示例
UNIXPATH	匹配 URL 路径，可能包含查询参数。	20	输入：/search?q=regex 模式：%{UNIXPATH:path}

Grok 模式	说明	最大模式限制	示例
			输出：{"path":"/search?q=regex"}
WINPATH	匹配 Windows 文件路径。	5	输入：C:\Users\John\Documents\file.txt 模式：%{WINPATH:path} 输出：{"path": "C:\\Users\\John\\Documents\\file.txt"}
路径	匹配 URL 或 Windows 文件路径	5	输入：/search?q=regex 模式：%{PATH:path} 输出：{"path":"/search?q=regex"}
TTY	匹配终端和伪终端的 Unix 设备路径。	20	输入：/dev/tty1 模式：%{TTY:path} 输出：{"path":"/dev/tty1"}
URIPROTO	匹配字母，可后跟加号（+）字符和其他字母或加号（+）字符	20	输入：web+transformer 模式：%{URIPROTO:protocol} 输出：{"protocol":"web+transformer"}

Grok 模式	说明	最大模式限制	示例
URIPATH	匹配 URI 的路径组件	20	输入：/category/sub-category/product_name 模式：%{URIPATH:path} 输出：{"path":"/category/sub-category/product_name"}
URIPARAM	匹配 URL 查询参数	5	输入：?param1=value1&param2=value2 模式：%{URIPARAM:url} 输出：{"url":"?param1=value1&param2=value2"}
URIPATHPARAM	匹配 URI 路径，可后跟查询参数	5	输入：/category/sub-category/product?id=12345&color=red 模式：%{URIPATHPARAM:path} 输出：{"path":"/category/sub-category/product?id=12345&color=red"}

Grok 模式	说明	最大模式限制	示例
URI	匹配完整的 URI	5	输入：`https://user:password@example.com/path/to/resource?param1=value1&param2=value2` 模式：`%{URI:uri}` 输出：`{"path":"https://user:password@example.com/path/to/resource?param1=value1&param2=value2"}`

日期和时间 grok 模式

Grok 模式	说明	最大模式限制	示例
MONTH	将完整的或缩写的英文月份名称匹配为完整单词	20	输入：`Jan` 模式：`%{MONTH:month}` 输出：`{"month":"Jan"}` 输入：`January` 模式：`%{MONTH:month}` 输出：`{"month":"January"}`

Grok 模式	说明	最大模式限制	示例
MONTHNUM	匹配 1 到 12 之间的月份数字，个位数月份可选择使用前导零。	20	输入：5 模式：%{MONTHNUM:month} 输出：{"month":"5"} 输入：05 模式：%{MONTHNUM:month} 输出：{"month":"05"}
MONTHNUM	匹配 01 到 12 之间的两位数月份数字。	20	输入：05 模式：%{MONTHNUM2:month} 输出：{"month":"05"}
MONTHDAY	匹配月份中的日期，从 1 到 31，可选择使用前导零。	20	输入：31 模式：%{MONTHDAY:monthDay} 输出：{"monthDay":"31"}

Grok 模式	说明	最大模式限制	示例
YEAR	匹配两位或四位数的年份	20	输入：2024 模式：%{YEAR:year} 输出：{"year":"2024"} 输入：24 模式：%{YEAR:year} 输出：{"year":"24"}
DAY	匹配完整或缩写的日期名称。	20	输入：Tuesday 模式：%{DAY:day} 输出：{"day":"Tuesday"}
HOUR	以 24 小时制格式匹配小时，可选择使用前导零 (0)0-23。	20	输入：22 模式：%{HOUR:hour} 输出：{"hour":"22"}
MINUTE	匹配分钟（00-59）。	20	输入：59 模式：%{MINUTE:min} 输出：{"min":"59"}

Grok 模式	说明	最大模式限制	示例
SECOND	匹配代表秒 (0)0-60 的数字，可后跟小数点或冒号，以及一个或多个数字表示小数分钟	20	输入：3 模式：%{SECOND:second} 输出：{"second":"3"} 输入：30.5 模式：%{SECOND:minSec} 输出：{"minSec":"30.5"} 输入：30:5 模式：%{SECOND:minSec} 输出：{"minSec":"30:5"}
TIME	将时间格式与小时、分钟和秒匹配，格式为 (H)H:mm:(s)s。秒包括闰秒 (0)0-60。	20	输入：09:45:32 模式：%{TIME:time} 输出：{"time":"09:45:32"}

Grok 模式	说明	最大模式限制	示例
DATE_US	以 (M)M/(d)d/(yy)yy 或 (M)M-(d)d-(yy)yy 格式匹配日期。	20	输入：11/23/2024 模式：%{DATE_US:date} 输出：{"date":"11/23/2024"} 输入：1-01-24 模式：%{DATE_US:date} 输出：{"date":"1-01-24"}
DATE_EU	以 (d)d/(M)M/(yy)yy、(d)d-(M)M-(yy)yy 或 (d)d.(M)M.(yy)yy 格式匹配日期。	20	输入：23/11/2024 模式：%{DATE_EU:date} 输出：{"date":"23/11/2024"} 输入：1.01.24 模式：%{DATE_EU:date} 输出：{"date":"1.01.24"}

Grok 模式	说明	最大模式限制	示例
ISO8601_时区	将 UTC 偏移量“Z”或时区偏移量与可选冒号进行匹配，格式为 [+-](H)H(:)mm。	20	输入：+05:30 模式：%{ISO8601_TIMEZONE:tz} 输出：{"tz":"+05:30"} 输入：-530 模式：%{ISO8601_TIMEZONE:tz} 输出：{"tz":"-530"} 输入：Z 模式：%{ISO8601_TIMEZONE:tz} 输出：{"tz":"Z"}
ISO8601_SECOND	匹配代表秒 (0)0-60 的数字，可后跟小数点或冒号，以及一个或多个数字表示小数秒	20	输入：60 模式：%{ISO8601_SECOND:second} 输出：{"second":"60"}

Grok 模式	说明	最大模式限制	示例
时间戳_ISO8601	将 ISO8601 日期时间格式 (yy) yy-(M) M-(d) dT (H) H: mm: (s) (Z| [+-] (H) H: mm) 与可选的秒数和时区进行匹配。	20	输入：2023-05-15T14:30:00+05:30 模式：%{TIMESTAMP_ISO8601:timestamp} 输出：{"timestamp":"2023-05-15T14:30:00+05:30"} 输入：23-5-1T1:25+5:30 模式：%{TIMESTAMP_ISO8601:timestamp} 输出：{"timestamp":"23-5-1T1:25+5:30"} 输入：23-5-1T1:25Z 模式：%{TIMESTAMP_ISO8601:timestamp} 输出：{"timestamp":"23-5-1T1:25Z"}

Grok 模式	说明	最大模式限制	示例
DATE	使用 %{DATE_US} 匹配美国格式的日期，或者使用 %{DATE_EU} 匹配欧盟格式的日期	20	输入：11/29/2024 模式：%{DATE:date} 输出：{"date":"11/29/2024"} 输入：29.11.2024 模式：%{DATE:date} 输出：{"date":"29.11.2024"}
DATESTAMP	匹配 %{DATE}，后跟 %{TIME} 模式，用空格或连字符分隔。	20	输入：29-11-2024 14:30:00 模式：%{DATESTAMP:dateTime} 输出：{"dateTime":"29-11-2024 14:30:00"}
TZ	匹配常见的时区缩写（PST、PDT、MST、MDT、CST CDT、EST、EDT、UTC）。	20	输入：PDT 模式：%{TZ:tz} 输出：{"tz":"PDT"}

Grok 模式	说明	最大模式限制	示例
日期戳_RFC822	按以下格式匹配日期和时间：日 MonthName (D) D (YY) YY (H) H: mm: (s) s 时区	20	输入：Monday Jan 5 23 1:30:00 CDT 模式：%{DATESTAMP_RFC822:dateTime} 输出：{"dateTime":"Monday Jan 5 23 1:30:00 CDT"} 输入：Mon January 15 2023 14:30:00 PST 模式：%{DATESTAMP_RFC822:dateTime} 输出：{"dateTime":"Mon January 15 2023 14:30:00 PST"}

Grok 模式	说明	最大模式限制	示例
日期戳_RFC2822	匹配 RFC2822 日期时间格式：日，(d) d (yy) yy MonthName (H) H: mm: (s) s Z| [+-] (H) H: mm	20	输入：Mon, 15 May 2023 14:30:00 +0530 模式：%{DATESTAMP_RFC2822:dateTime} 输出：{"dateTime":"Mon, 15 May 2023 14:30:00 +0530"} 输入：Monday, 15 Jan 23 14:30:00 Z 模式：%{DATESTAMP_RFC2822:dateTime} 输出：{"dateTime":"Monday, 15 Jan 23 14:30:00 Z"}
DATESTAMP_OTHER	按格式匹配日期和时间：日 MonthName (d) d (H) H: mm: (s) s 时区 (yy) yy	20	输入：Mon May 15 14:30:00 PST 2023 模式：%{DATESTAMP_OTHER:dateTime} 输出：{"dateTime":"Mon May 15 14:30:00 PST 2023"}

Grok 模式	说明	最大模式限制	示例
DATESTAMP_EVENTLOG	匹配不带分隔符的紧凑日期时间格式：(yy)yyMM(d)d(H)Hmm(s)s	20	输入：20230515143000 模式：%{DATESTAMP_EVENTLOG:dateTime} 输出：{"dateTime":"20230515143000"}

日志 grok 模式

Grok 模式	说明	最大模式限制	示例
LOGLEVEL	匹配不同大小写和缩写形式的标准日志级别，包括：Alert/ALERT 、Trace/TRACE 、Debug/DEBUG 、Notice/NOTICE 、Info/INFO 、Warn/Warning/WARN/WARNING 、Err/Error/ERR/ERROR 、Crit/Critical/CRIT/CRITICAL 、Fatal/FATAL 、Severe/SEVERE 、Emerg/Emergency/EMERG/EMERGENCY	20	输入：INFO 模式：%{LOGLEVEL:logLevel} 输出：{"logLevel":"INFO"}
HTTPDATE	匹配日志文件中经常使用的日期和时间格式。格	20	输入：23/Nov/2024:14:30:00 +0640

Grok 模式	说明	最大模式限制	示例
	式：(d) d/MonthName/(yy) yy :(H) H: mm: (s) s 时区：匹配完整或缩写的英文月份名称（例如 MonthName：“一月”或“一月”）时区：匹配%{INT} grok 模式		模式：%{HTTPDATE:date} 输出：{"date":"23/Nov/2024:14:30:00 +0640"}
SYSLOGTIMESTAMP	将日期格式与 MonthName (d) d (H) H: mm :(s) s 匹配完整或缩写的英文月份名称（例如 MonthName：“Jan”或“January”）	20	输入：Nov 29 14:30:00 模式：%{SYSLOGTIMESTAMP:dateTime} 输出：{"dateTime":"Nov 29 14:30:00"}
PROG	匹配由字母、数字、点、下划线、正斜杠、百分号和连字符组成的程序名称。	20	输入：user.profile/settings-page 模式：%{PROG:program} 输出：{"program":"user.profile/settings-page"}

Grok 模式	说明	最大模式限制	示例
SYSLOGPRO G	匹配 PROG grok 模式，可后跟方括号中的进程 ID。	20	输入：user.profile/settings-page[1234] 模式：%{SYSLOGPROG:programWithId} 输出：{"programWithId":"user.profile/settings-page[1234]","program":"user.profile/settings-page","pid":"1234"}
SYSLOGH	匹配 %{HOST} 或 %{IP} 模式	5	输入：2001:db8:3333:4444:5555:6666:7777:8888 模式：%{SYSLOGHOST:ip} 输出：{"ip":"2001:db8:3333:4444:5555:6666:7777:8888"}

Grok 模式	说明	最大模式限制	示例
SYSLOGFAC ILITY	匹配十进制格式的 syslog 优先级。该值应括在尖括号（<>）中。	20	输入：<13.6> 模式：%{SYSLOGF ACILITY:syslog} 输出：{"syslog" :"<13.6>" ,"facilit y":"13"," priority":"6"}

常见的 log grok 模式

您可以使用预定义的自定义 grok 模式来匹配 Apache、NGINX 和 Syslog 协议（RFC 5424）日志格式。使用这些特定模式时，它们必须是匹配配置中的第一个模式，并且不能有其他模式位于它们之前。此外，后面只有跟有一个 DATA。GREEDYDATA 或 GREEDYDATA_MULTILINE 模式。

Grok 模式	说明	最大模式限制
APACHE_ACCESS_LOG	匹配 Apache 访问日志	1
NGINX_ACCESS_LOG	匹配 NGINX 访问日志	1
SYSLOG5424	匹配 Syslog 协议（RFC 5424）日志	1

下面展示了使用这些常见日志格式模式的有效和无效示例。

```
"%{NGINX_ACCESS_LOG} %{DATA}" // Valid
"%{SYSLOG5424}%{DATA:logMsg}" // Valid
"%{APACHE_ACCESS_LOG} %{GREEDYDATA:logMsg}" // Valid
"%{APACHE_ACCESS_LOG} %{SYSLOG5424}" // Invalid (multiple common log patterns used)
```

```
"%{NGINX_ACCESS_LOG} %{NUMBER:num}" // Invalid (Only GREEDYDATA and DATA patterns are
 supported with common log patterns)
"%{GREEDYDATA:logMsg} %{SYSLOG5424}" // Invalid (GREEDYDATA and DATA patterns are
 supported only after common log patterns)
```

常见日志格式示例

Apache 日志示例

日志示例：

```
127.0.0.1 - - [03/Aug/2023:12:34:56 +0000] "GET /page.html HTTP/1.1" 200 1234
```

转换器：

```
[
     {
        "grok": {
            "match": "%{APACHE_ACCESS_LOG}"
        }
    }
]
```

输出：

```
{
    "request": "/page.html",
    "http_method": "GET",
    "status_code": 200,
    "http_version": "1.1",
    "response_size": 1234,
    "remote_host": "127.0.0.1",
    "timestamp": "2023-08-03T12:34:56Z"
}
```

NGINX 日志示例

日志示例：

```
192.168.1.100 - Foo [03/Aug/2023:12:34:56 +0000] "GET /account/login.html HTTP/1.1"
 200 42 "https://www.amazon.com/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
```

转换器：

```
[
     {
        "grok": {
            "match": "%{NGINX_ACCESS_LOG}"
        }
    }
]
```

输出：

```
{
    "request": "/account/login.html",
    "referrer": "https://www.amazon.com/",
    "agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/92.0.4515.131 Safari/537.36",
    "http_method": "GET",
    "status_code": 200,
    "auth_user": "Foo",
    "http_version": "1.1",
    "response_size": 42,
    "remote_host": "192.168.1.100",
    "timestamp": "2023-08-03T12:34:56Z"
}
```

Syslog 协议（RFC 5424）日志示例

日志示例：

```
<165>1 2003-10-11T22:14:15.003Z mymachine.example.com evntslog - ID47
 [exampleSDID@32473 iut="3" eventSource= "Application" eventID="1011"]
[examplePriority@32473 class="high"]
```

转换器：

```
[
     {
        "grok": {
            "match": "%{SYSLOG5424}"
        }
    }
```

```
]
```

输出：

```
{
  "pri": 165,
  "version": 1,
  "timestamp": "2003-10-11T22:14:15.003Z",
  "hostname": "mymachine.example.com",
  "app": "evntslog",
  "msg_id": "ID47",
  "structured_data": "exampleSDID@32473 iut=\"3\" eventSource= \"Application\" eventID=
\"1011\"",
  "message": "[examplePriority@32473 class=\"high\"]"
}
```

CSV

csv 处理器将日志事件中的逗号分隔值（CSV）解析为列。

字段	说明	必填？	默认	限制
source	日志事件中待解析字段的路径	否	`@message`	最大长度：128 最大嵌套键深度：3
分隔符	用于分隔原始逗号分隔值日志事件中的每列的字符	否	`,`	最大长度：1，除非值为 `\t` 或 `\s`
quoteCharacter	用作单列数据的文本限定符的字符	否	`"`	最大长度：1
列	转换后的日志事件中列要使用的名称列表。	否	`[column_1, column_2`	CSV 最大列数：100 最大长度：128 最大嵌套键深度：3
destination	放置转换后的键值对的父字段	否	`Root node`	最大长度：128 最大嵌套键深度：3

将 delimiter 设置为 \t 将使用制表符分隔每列，\t 将使用单个空格字符分隔每列。

示例

假设摄取的部分日志事件如下所示：

```
'Akua Mansa':28:'New York: USA'
```

假设我们仅使用 csv 处理器：

```
[
     {
        "csv": {
            "delimiter": ":",
            "quoteCharacter": "'"
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
  "column_1": "Akua Mansa",
  "column_2": "28",
  "column_3": "New York: USA"
}
```

示例 2

假设摄取的日志事件如下所示：

```
{
    "timestamp": "2024-11-23T16:03:12Z",
    "type": "user_data",
    "logMsg": "'Akua Mansa':28:'New York: USA'"
}
```

假设我们将事件解析为 JSON，他们使用 cs v 处理器解析一个 JSON 字段，指定列名和目标：

```
[
    {
        "parseJSON": {}
```

```
    },
    {
        "csv": {
            "source": "logMsg",
            "delimiter": ":",
            "quoteCharacter": "'",
            "columns":["name","age","location"],
            "destination": "msg"
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
    "timestamp": "2024-11-23T16:03:12Z",
    "logMsg": "'Akua Mansa':28:'New York: USA'",
    "type": "user_data",
    "msg": {
        "name": "Akua Mansa",
        "age": "28",
        "location": "New York: USA"
    }
}
```

parseKeyValue

使用parseKeyValue处理器将指定字段解析为键值对。您可以使用以下选项自定义处理器以解析字段信息。

字段	说明	必填？	默认	限制
source	日志事件中待解析字段的路径	否	`@message`	最大长度：128 最大嵌套键深度：3
destination	要将提取的键值对放入的目标字段	否		最大长度：128
fieldDelimiter	在原始日志事件的键值对之间使用的字段分隔符字符串	否	&	最大长度：128

字段	说明	必填？	默认	限制
keyValueDelimiter	在转换后的日志事件中每对的键和值之间使用的分隔符字符串	否	=	最大长度：128
nonMatchValue	当键值对未成功拆分时要在结果的值字段中插入的值。	否		最大长度：128
keyPrefix	如果要为所有转换后的键添加前缀，请在此处指定。	否		最大长度：128
overwriteIfExists	如果目标键已存在，是否覆盖该键	否	false	

示例

以以下日志事件为例：

```
key1:value1!key2:value2!key3:value3!key4
```

假设我们使用下面的处理器配置：

```
[
    {
        "parseKeyValue": {
            "destination": "new_key",
            "fieldDelimiter": "!",
            "keyValueDelimiter": ":",
            "nonMatchValue": "defaultValue",
            "keyPrefix": "parsed_"
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
  "new_key": {
    "parsed_key1": "value1",
    "parsed_key2": "value2",
```

```
    "parsed_key3": "value3",
    "parsed_key4": "defaultValue"
  }
}
```

用于 AWS 出售日志的内置处理器

本节包含有关内置处理器的信息，您可以将这些处理器与 AWS 提供日志的服务一起使用。

目录

- parseWAF
- parsePostgres
- parseCloudfront
- parseRoute53
- parseVPC
- parseToOCSF

parseWAF

使用此处理器来解析 AWS WAF 出售的日志，它获取每个标头名称的内容`httpRequest.headers`并使用相应的值创建 JSON 密钥。它还会对 `labels` 执行相同的操作。这些转换可以使查询 AWS WAF 日志变得更加容易。有关 AWS WAF 日志格式的更多信息，请参阅 Web ACL 流量的日志示例。

此处理器仅接受 `@message` 作为输入。

> **Important**
>
> 如果您使用此处理器，则它必须是转换器中的第一个处理器。

示例

以以下日志事件为例：

```
{
  "timestamp": 1576280412771,
  "formatVersion": 1,
```

```
  "webaclId": "arn:aws:wafv2:ap-southeast-2:111122223333:regional/webacl/
STMTest/1EXAMPLE-2ARN-3ARN-4ARN-123456EXAMPLE",
  "terminatingRuleId": "STMTest_SQLi_XSS",
  "terminatingRuleType": "REGULAR",
  "action": "BLOCK",
  "terminatingRuleMatchDetails": [
    {
      "conditionType": "SQL_INJECTION",
      "sensitivityLevel": "HIGH",
      "location": "HEADER",
      "matchedData": ["10", "AND", "1"]
    }
  ],
  "httpSourceName": "-",
  "httpSourceId": "-",
  "ruleGroupList": [],
  "rateBasedRuleList": [],
  "nonTerminatingMatchingRules": [],
  "httpRequest": {
    "clientIp": "1.1.1.1",
    "country": "AU",
    "headers": [
      { "name": "Host", "value": "localhost:1989" },
      { "name": "User-Agent", "value": "curl/7.61.1" },
      { "name": "Accept", "value": "*/*" },
      { "name": "x-stm-test", "value": "10 AND 1=1" }
    ],
    "uri": "/myUri",
    "args": "",
    "httpVersion": "HTTP/1.1",
    "httpMethod": "GET",
    "requestId": "rid"
  },
  "labels": [{ "name": "value" }]
}
```

处理器配置如下所示：

```
[
    {
        "parseWAF": {}
    }
]
```

转换后的日志事件将如下所示。

```
{
  "httpRequest": {
    "headers": {
      "Host": "localhost:1989",
      "User-Agent": "curl/7.61.1",
      "Accept": "*/*",
      "x-stm-test": "10 AND 1=1"
    },
    "clientIp": "1.1.1.1",
    "country": "AU",
    "uri": "/myUri",
    "args": "",
    "httpVersion": "HTTP/1.1",
    "httpMethod": "GET",
    "requestId": "rid"
  },
  "labels": { "name": "value" },
  "timestamp": 1576280412771,
  "formatVersion": 1,
  "webaclId": "arn:aws:wafv2:ap-southeast-2:111122223333:regional/webacl/
STMTest/1EXAMPLE-2ARN-3ARN-4ARN-123456EXAMPLE",
  "terminatingRuleId": "STMTest_SQLi_XSS",
  "terminatingRuleType": "REGULAR",
  "action": "BLOCK",
  "terminatingRuleMatchDetails": [
    {
      "conditionType": "SQL_INJECTION",
      "sensitivityLevel": "HIGH",
      "location": "HEADER",
      "matchedData": ["10", "AND", "1"]
    }
  ],
  "httpSourceName": "-",
  "httpSourceId": "-",
  "ruleGroupList": [],
  "rateBasedRuleList": [],
  "nonTerminatingMatchingRules": []
}
```

parsePostgres

使用此处理器来解析 Amazon RDS for PostgreSQL 出售的日志、提取字段并将其转换为 JSON 格式。有关 RDS for PostgreSQL 日志格式的更多信息，请参阅 RDS for PostgreSQL 据库日志文件。

此处理器仅接受 `@message` 作为输入。

> **Important**
>
> 如果您使用此处理器，则它必须是转换器中的第一个处理器。

示例

以以下日志事件为例：

```
2019-03-10 03:54:59 UTC:10.0.0.123(52834):postgres@logtestdb:[20175]:ERROR: column
 "wrong_column_name" does not exist at character 8
```

处理器配置如下所示：

```
[
    {
        "parsePostgres": {}
    }
]
```

转换后的日志事件将如下所示。

```
{
  "logTime": "2019-03-10 03:54:59 UTC",
  "srcIp": "10.0.0.123(52834)",
  "userName": "postgres",
  "dbName": "logtestdb",
  "processId": "20175",
  "logLevel": "ERROR"
}
```

parseCloudfront

使用此处理器来解析 Amazon CloudFront 出售的日志、提取字段并将其转换为 JSON 格式。对编码的字段值进行解码。整数和双精度值按此方法处理。有关 Amazon CloudFront 日志格式的更多信息，请参阅配置和使用标准日志（访问日志）。

此处理器仅接受 `@message` 作为输入。

> **Important**
>
> 如果您使用此处理器，则它必须是转换器中的第一个处理器。

示例

以以下日志事件为例：

```
2019-12-04  21:02:31   LAX1   392    192.0.2.24    GET
 d111111abcdef8.cloudfront.net  /index.html    200    -  Mozilla/5.0%20(Windows
%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,
%20like%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36  -  -  Hit
    SOX4xwn4XV6Q4rgb7XiVGOHms_BGlTAC4KyHmureZmBNrjGdRLiNIQ==
 d111111abcdef8.cloudfront.net  https  23 0.001  -  TLSv1.2    ECDHE-RSA-AES128-GCM-
SHA256    Hit    HTTP/2.0   -  -  11040  0.001  Hit    text/html  78 -  -
```

处理器配置如下所示：

```
[
    {
        "parseCloudfront": {}
    }
]
```

转换后的日志事件将如下所示。

```
{
  "date": "2019-12-04",
  "time": "21:02:31",
  "x-edge-location": "LAX1",
  "sc-bytes": 392,
  "c-ip": "192.0.2.24",
  "cs-method": "GET",
```

```
  "cs(Host)": "d111111abcdef8.cloudfront.net",
  "cs-uri-stem": "/index.html",
  "sc-status": 200,
  "cs(Referer)": "-",
  "cs(User-Agent)": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36",
  "cs-uri-query": "-",
  "cs(Cookie)": "-",
  "x-edge-result-type": "Hit",
  "x-edge-request-id": "SOX4xwn4XV6Q4rgb7XiVGOHms_BGlTAC4KyHmureZmBNrjGdRLiNIQ==",
  "x-host-header": "d111111abcdef8.cloudfront.net",
  "cs-protocol": "https",
  "cs-bytes": 23,
  "time-taken": 0.001,
  "x-forwarded-for": "-",
  "ssl-protocol": "TLSv1.2",
  "ssl-cipher": "ECDHE-RSA-AES128-GCM-SHA256",
  "x-edge-response-result-type": "Hit",
  "cs-protocol-version": "HTTP/2.0",
  "fle-status": "-",
  "fle-encrypted-fields": "-",
  "c-port": 11040,
  "time-to-first-byte": 0.001,
  "x-edge-detailed-result-type": "Hit",
  "sc-content-type": "text/html",
  "sc-content-len": 78,
  "sc-range-start": "-",
  "sc-range-end": "-"
}
```

parseRoute53

使用此处理器来解析 Amazon Route 53 Public Data Plane 出售的日志、提取字段并将其转换为 JSON 格式。对编码的字段值进行解码。此处理器不支持 Amazon Route 53 Resolver 日志。

此处理器仅接受 `@message` 作为输入。

> **Important**
>
> 如果您使用此处理器，则它必须是转换器中的第一个处理器。

示例

以以下日志事件为例：

```
1.0 2017-12-13T08:15:50.235Z Z123412341234 example.com AAAA NOERROR TCP IAD12 192.0.2.0
 198.51.100.0/24
```

处理器配置如下所示：

```
[
    {
        "parseRoute53": {}
    }
]
```

转换后的日志事件将如下所示。

```
{
  "version": 1.0,
  "queryTimestamp": "2017-12-13T08:15:50.235Z",
  "hostZoneId": "Z123412341234",
  "queryName": "example.com",
  "queryType": "AAAA",
  "responseCode": "NOERROR",
  "protocol": "TCP",
  "edgeLocation": "IAD12",
  "resolverIp": "192.0.2.0",
  "ednsClientSubnet": "198.51.100.0/24"
}
```

parseVPC

使用此处理器来解析 Amazon VPC 公开发布的日志、提取字段并将其转换为 JSON 格式。对编码的字段值进行解码。

此处理器仅接受 `@message` 作为输入。

> **Important**
>
> 如果您使用此处理器，则它必须是转换器中的第一个处理器。

示例

以以下日志事件为例：

```
2 123456789010 eni-abc123de 192.0.2.0 192.0.2.24 20641 22 6 20 4249 1418530010
 1418530070 ACCEPT OK
```

处理器配置如下所示：

```
[
    {
        "parseVPC": {}
    }
]
```

转换后的日志事件将如下所示。

```
{
  "version": 2,
  "accountId": "123456789010",
  "interfaceId": "eni-abc123de",
  "srcAddr": "192.0.2.0",
  "dstAddr": "192.0.2.24",
  "srcPort": 20641,
  "dstPort": 22,
  "protocol": 6,
  "packets": 20,
  "bytes": 4249,
  "start": 1418530010,
  "end": 1418530070,
  "action": "ACCEPT",
  "logStatus": "OK"
}
```

parseToOCSF

parseToOCSF 处理器将日志转换为开放式网络安全架构框架（OCSF）事件。OCSF 是一种开放标准，它为安全数据提供了通用架构，从而能够在不同安全工具和平台之间实现更好的互操作性和分析。

该处理器对于安全分析工作流程特别有用，在这些工作流程中，您需要将来自各种 AWS 服务的日志格式标准化为一致的架构以进行下游分析。

参数

eventSource（必需）

指定生成要转换的日志事件的 AWS 服务或进程。有效值为：

- CloudTrail- CloudTrail 日志
- Route53Resolver：Route 53 Resolver 日志
- VPCFlow：Amazon VPC 流日志
- EKSAudit：Amazon EKS 审计日志
- AWSWAF- AWS WAF 日志

ocsfVersion（必需）

指定用于转换后的日志事件的 OCSF 架构版本。目前支持的版本：V1.1, V1.5

mappingVersion（可选）

指定 OCSF 转换映射版本。控制在将日志转换为 OCSF 格式时应用哪种转换逻辑。如果未指定，则在创建策略时使用最新的可用版本。发布新的映射版本时，现有策略不会自动升级。当前最新版本:v1.5.0.

注意：在支持时ocsfVersion不支持V1.1。

source（可选）

要解析的日志事件中字段的路径。如果省略，则解析整条日志消息。

示例

以下示例展示了如何使用 parseToOCSF 将 VPC 流日志转换为 OCSF 格式：

```
{
  "parseToOCSF": {
    "eventSource": "VPCFlow",
    "ocsfVersion": "V1.1"
  }
}
```

以下示例说明如何为一致的转换行为指定特定的映射版本：

```
{
  "parseToOCSF": {
    "eventSource": "CloudTrail",
    "ocsfVersion": "V1.5",
```

```
    "mappingVersion": "v1.5.0"
  }
}
```

字符串变异处理器

本节包含有关可与日志事件转换器配合使用的字符串变异处理器的信息。

目录

- lowerCaseString
- upperCaseString
- splitString
- substituteString
- trimString

lowerCaseString

`lowerCaseString` 处理器会将字符串转换为其小写版本。

字段	说明	必填？	默认	限制
withKeys	要转换为小写的键列表	是		最多条目数：10

示例

以以下日志事件为例：

```
{
    "outer_key": {
        "inner_key": "INNER_VALUE"
    }
}
```

转换器配置如下，将 `lowerCaseString` 和 `parseJSON` 结合使用：

```
[
    {
```

```
        "parseJSON": {}
    },
    {
        "lowerCaseString": {
            "withKeys":["outer_key.inner_key"]
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

upperCaseString

upperCaseString 处理器会将字符串转换为其大写版本。

字段	说明	必填？	默认	限制
withKeys	要转换为大写的键列表	是		最多条目数：10

示例

以以下日志事件为例：

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

转换器配置如下，将 upperCaseString 和 parseJSON 结合使用：

```
[
    {
        "parseJSON": {}
```

```
    },
    {
        "upperCaseString": {
            "withKeys":["outer_key.inner_key"]
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": "INNER_VALUE"
  }
}
```

splitString

`splitString` 处理器是一种字符串变异处理器，它使用分隔字符将字段拆分为数组。

字段	说明	必填？	默认	限制
entries	条目数组。数组中的每个项目都必须包含 `source` 和 `delimiter` 字段。	是		最多条目数：10
source	要拆分的字段值的键	是		最大长度：128
分隔符	用于拆分字段值的分隔符字符串	是		最大长度：128

示例 1

以以下日志事件为例：

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

转换器配置如下，将 splitString 和 parseJSON 结合使用：

```
[
     {
        "parseJSON": {}
    },
    {
         "splitString": {
            "entries": [
                {
                    "source": "outer_key.inner_key",
                    "delimiter": "_"
                }
            ]
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": [
      "inner",
      "value"
    ]
  }
}
```

示例 2

用于拆分字符串的分隔符可以包含多个字符。

以以下日志事件为例：

```
{
    "outer_key": {
        "inner_key": "item1, item2, item3"
    }
}
```

转换器配置如下：

```
[
     {
        "parseJSON": {}
    },
    {
         "splitString": {
            "entries": [
                {
                    "source": "outer_key.inner_key",
                    "delimiter": ", "
                }
            ]
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": [
      "item1",
      "item2",
      "item3"
    ]
  }
}
```

substituteString

substituteString 处理器是一种字符串变异处理器，它将键的值与正则表达式进行匹配，并将所有匹配项替换为替换字符串。

字段	说明	必填？	默认	限制
entries	条目数组。数组中的每个项目都必须包含 source、from 和 to 字段。	是		最多条目数：10
source	要修改的字段的键	是		最大长度：128

字段	说明	必填？	默认	限制
				最大嵌套键深度：3
from	要替换的正则表达式字符串。特殊正则表达式字符（例如 [和]）在使用双引号时必须使用 \\ 进行转义，在使用单引号时或通过 AWS 管理控制台配置时必须使用 \ 进行转义。有关更多信息，请参阅 Oracle 网站上的 Class Pattern。 您可以将模式封装在 `(...)` 中以创建带编号的捕获组，并创建可在 to 字段中引用的 `(?P<group_name>...)` 命名捕获组。	是		最大长度：128
修改为	对于捕获组的 from 反向引用的每个匹配项，可以使用要替换的字符串。对编号组使用 $n 形式（如 $1），对命名组使用 `${group_name}`（如 `${my_group}`）。	是		最大长度：128 反向引用最大数量：10 重复反向引用最大数量：2

示例 1

以以下日志事件为例：

```
{
    "outer_key": {
        "inner_key1": "[]",
        "inner_key2": "123-345-567",
        "inner_key3": "A cat takes a catnap."
    }
}
```

转换器配置如下，将 substituteString 和 parseJSON 结合使用：

```
[
```

```
    {
        "parseJSON": {}
    },
    {
        "substituteString": {
            "entries": [
                {
                    "source": "outer_key.inner_key1",
                    "from": "\\[\\]",
                    "to": "value1"
                },
                {
                    "source": "outer_key.inner_key2",
                    "from": "[0-9]{3}-[0-9]{3}-[0-9]{3}",
                    "to": "xxx-xxx-xxx"
                },
                {
                    "source": "outer_key.inner_key3",
                    "from": "cat",
                    "to": "dog"
                }
            ]
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key1": "value1",
    "inner_key2": "xxx-xxx-xxx",
    "inner_key3": "A dog takes a dognap."
  }
}
```

示例 2

以以下日志事件为例：

```
{
    "outer_key": {
        "inner_key1": "Tom, Dick, and Harry",
```

```
        "inner_key2": "arn:aws:sts::123456789012:assumed-role/MyImportantRole/
MySession"
    }
}
```

转换器配置如下，将 substituteString 和 parseJSON 结合使用：

```
[
    {
        "parseJSON": {}
    },
    {
        "substituteString": {
            "entries": [
                {
                    "source": "outer_key.inner_key1",
                    "from": "(\w+), (\w+), and (\w+)",
                    "to": "$1 and $3"
                },
                {
                    "source": "outer_key.inner_key2",
                    "from": "^arn:aws:sts::(?P<account_id>\\d{12}):assumed-role/(?
P<role_name>[\\w+=,.@-]+)/(?P<role_session_name>[\\w+=,.@-]+)$",
                    "to": "${account_id}:${role_name}:${role_session_name}"
                }
            ]
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key1": "Tom and Harry",
    "inner_key2": "123456789012:MyImportantRole:MySession"
  }
}
```

trimString

trimString 处理器会移除键开头和结尾的空格。

字段	说明	必填？	默认	限制
withKeys	要剪裁的键列表	是		最多条目数：10

示例

以以下日志事件为例：

```
{
    "outer_key": {
        "inner_key": "   inner_value  "
    }
}
```

转换器配置如下，将 `trimString` 和 `parseJSON` 结合使用：

```
[
    {
        "parseJSON": {}
    },
    {
        "trimString": {
            "withKeys":["outer_key.inner_key"]
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

JSON 变异处理器

本节包含有关可用于日志事件转换器的 JSON 变异处理器的信息。

目录

- addKeys
- deleteKeys
- moveKeys
- renameKeys
- copyValue
- listToMap

addKeys

使用 addKeys 处理器向日志事件添加新的键值对。

字段	说明	必填？	默认	限制
entries	条目数组。数组中的每个项目都可以包含 key、value 和 overwriteIfExists 字段。	是		最多条目数：5
键	要添加的新条目的键	是		最大长度：128 最大嵌套键深度：3
值	要添加的新条目的值	是		最大长度：256
overwriteIfExists	如果您将其设置为 true，而事件中已存在 key，则现有值会被覆盖。默认值为 false。	否	false	无限制

示例

以以下日志事件为例：

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

转换器配置如下，将 addKeys 和 parseJSON 结合使用：

```
[
    {
        "parseJSON": {}
    },
    {
        "addKeys": {
            "entries": [
                {
                    "key": "outer_key.new_key",
                    "value": "new_value"
                }
            ]
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": "inner_value",
    "new_key": "new_value"
  }
}
```

deleteKeys

使用 deleteKeys 处理器从日志事件中删除字段。这些字段可以包含键值对。

字段	说明	必填？	默认	限制
withKeys	要删除的键列表。	是	无限制	最多条目数：5

示例

以以下日志事件为例：

```
{
```

```
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

转换器配置如下，将 deleteKeys 和 parseJSON 结合使用：

```
[
    {
        "parseJSON": {}
    },
    {
        "deleteKeys": {
            "withKeys":["outer_key.inner_key"]
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {}
}
```

moveKeys

使用 moveKeys 处理器将键从一个字段移动到另一个字段。

字段	说明	必填？	默认	限制
entries	条目数组。数组中的每个项目都可以包含 source、target 和 overwriteIfExists 字段。	是		最多条目数：5
source	要移动的键	是		最大长度：128 最大嵌套键深度：3
target	要移动至的键	是		最大长度：128 最大嵌套键深度：3

字段	说明	必填？	默认	限制
overwriteIfExists	如果您将其设置为 `true`，而事件中已存在 `key`，则现有值会被覆盖。默认值为 `false`。	否	false	无限制

示例

以以下日志事件为例：

```
{
    "outer_key1": {
        "inner_key1": "inner_value1"
    },
    "outer_key2": {
        "inner_key2": "inner_value2"
    }
}
```

转换器配置如下，将 `moveKeys` 和 `parseJSON` 结合使用：

```
[
    {
        "parseJSON": {}
    },
    {
        "moveKeys": {
            "entries": [
                {
                    "source": "outer_key1.inner_key1",
                    "target": "outer_key2"
                }
            ]
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
```

```
  "outer_key1": {},
  "outer_key2": {
    "inner_key2": "inner_value2",
    "inner_key1": "inner_value1"
  }
}
```

renameKeys

使用 renameKeys 处理器重命名日志事件中的键。

字段	说明	必填？	默认	限制
entries	条目数组。数组中的每个项目都可以包含 key、target 和 overwriteIfExists 字段。	是	无限制	最多条目数：5
键	要重命名的键	是	无限制	最大长度：128
target	新的键名称	是	无限制	最大长度：128 最大嵌套键深度：3
overwriteIfExists	如果您将其设置为 true，而事件中已存在 key，则现有值会被覆盖。默认值为 false。	否	false	无限制

示例

以以下日志事件为例：

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

转换器配置如下，将 renameKeys 和 parseJSON 结合使用：

```
[
```

```
    {
        "parseJSON": {}
    },
    {
        "renameKeys": {
            "entries": [
                {
                    "key": "outer_key",
                    "target": "new_key"
                }
            ]
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
  "new_key": {
    "inner_key": "inner_value"
  }
}
```

copyValue

使用 copyValue 处理器复制日志事件中的值。您还可以通过将以下元数据键的值复制到日志事件中，来使用此处理器向日志事件添加元数据：@logGroupName、@logGroupStream、@accountId、@regionName。以下示例对此进行了说明。

字段	说明	必填？	默认	限制
entries	条目数组。数组中的每个项目都可以包含 source、target 和 overwriteIfExists 字段。	是		最多条目数：5
source	要复制的键	是		最大长度：128 最大嵌套键深度：3
target	将值复制到的键	是	无限制	最大长度：128

字段	说明	必填？	默认	限制
				最大嵌套键深度：3
overwriteIfExists	如果您将其设置为 `true`，而事件中已存在 `key`，则现有值会被覆盖。默认值为 `false`。	否	false	无限制

示例

以以下日志事件为例：

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

转换器配置如下，将 `copyValue` 和 `parseJSON` 结合使用：

```
[
    {
        "parseJSON": {}
    },
    {
        "copyValue": {
            "entries": [
                {
                    "key": "outer_key.new_key",
                    "target": "new_key"
                },
                {
                    "source": "@logGroupName",
                    "target": "log_group_name"
                },
                {
                    "source": "@logGroupStream",
                    "target": "log_group_stream"
                },
                {
                    "source": "@accountId",
                    "target": "account_id"
```

```
                },
                {
                    "source": "@regionName",
                    "target": "region_name"
                }
            ]
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
  "outer_key": {
    "inner_key": "inner_value"
  },
  "new_key": "inner_value",
  "log_group_name": "myLogGroupName",
  "log_group_stream": "myLogStreamName",
  "account_id": "012345678912",
  "region_name": "us-east-1"
}
```

listToMap

`listToMap` 处理器获取包含键字段的对象列表，并将其转换为目标键的映射。

字段	说明	必填？	默认	限制
source	中的密钥 ProcessingEvent 包含将要转换为地图的对象列表	是		最大长度：128 最大嵌套键深度：3
键	要在生成的映射中提取为键的字段的键	是		最大长度：128
valueKey	如果指定此项，则将从 `source` 对象中提取您在此参数中指定的值，并将其放入生成的映射的值中。否则，源列表中的原始对象将放入生成的映射的值中。	否		最大长度：128

字段	说明	必填？	默认	限制
target	将保存生成的映射的字段的键	否	根节点	最大长度：128 最大嵌套键深度：3
flatten	一个布尔值，指示列表是扁平化为单个项目，还是生成的映射中的值将是列表。 默认情况下，匹配键的值将以数组表示。请将 `flatten` 设置为 `true`，以根据 `flattened Element` 的值将数组转换为单个值。	否	false	
flattened Element	如果将 `flatten` 设置为 `true`，则使用 `flattenedElement` 指定要保留哪个元素（`first` 或 `last`）。	在 `flatten` 设置为 `true` 时是必需的		值只能是 `first` 或 `last`

示例

以以下日志事件为例：

```
{
    "outer_key": [
        {
            "inner_key": "a",
            "inner_value": "val-a"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b1"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b2"
        },
```

```
        {
            "inner_key": "c",
            "inner_value": "val-c"
        }
    ]
}
```

使用案例 1 的转换器：`flatten` 是 `false`

```
[
    {
        "parseJSON": {}
    },
    {
        "listToMap": {
            "source": "outer_key"
            "key": "inner_key",
            "valueKey": "inner_value",
            "flatten": false
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
    "outer_key": [
        {
            "inner_key": "a",
            "inner_value": "val-a"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b1"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b2"
        },
        {
            "inner_key": "c",
            "inner_value": "val-c"
        }
```

```
    ],
    "a": [
        "val-a"
    ],
    "b": [
        "val-b1",
        "val-b2"
    ],
    "c": [
        "val-c"
    ]
}
```

使用案例 2 的转换器：`flatten` 是 `true`，`flattenedElement` 是 `first`

```
[
    {
        "parseJSON": {}
    },
    {
        "listToMap": {
            "source": "outer_key"
            "key": "inner_key",
            "valueKey": "inner_value",
            "flatten": true,
            "flattenedElement": "first"
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
    "outer_key": [
        {
            "inner_key": "a",
            "inner_value": "val-a"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b1"
        },
        {
```

```
            "inner_key": "b",
            "inner_value": "val-b2"
        },
        {
            "inner_key": "c",
            "inner_value": "val-c"
        }
    ],
    "a": "val-a",
    "b": "val-b1",
    "c": "val-c"
}
```

使用案例 3 的转换器：`flatten` 是 `true`，`flattenedElement` 是 `last`

```
[
    {
        "parseJSON": {}
    },
    {
        "listToMap": {
            "source": "outer_key"
            "key": "inner_key",
            "valueKey": "inner_value",
            "flatten": true,
            "flattenedElement": "last"
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
    "outer_key": [
        {
            "inner_key": "a",
            "inner_value": "val-a"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b1"
        },
        {
```

```
            "inner_key": "b",
            "inner_value": "val-b2"
        },
        {
            "inner_key": "c",
            "inner_value": "val-c"
        }
    ],
    "a": "val-a",
    "b": "val-b2",
    "c": "val-c"
}
```

数据类型转换器处理器

本节包含有关可与日志事件转换器配合使用的数据类型转换器处理器的信息。

目录

- typeConverter
- datetimeConverter

typeConverter

使用 typeConverter 处理器可以将与指定的键关联的值类型转换为指定的类型。它是一个转换处理器，可以更改指定字段的类型。值可以转换为以下数据类型之一：integer、double、string 和 boolean。

字段	说明	必填？	默认	限制
entries	条目数组。数组中的每个项目都必须包含 key 和 type 字段。	是		最多条目数：10
键	要转换为其他类型的键和值	是		最大长度：128 最大嵌套键深度：3
类型	要转换为的类型。有效值包括 integer、double、string 和 boolean。	是		

示例

以以下日志事件为例：

```
{
    "name": "value",
    "status": "200"
}
```

转换器配置如下，将 typeConverter 和 parseJSON 结合使用：

```
[
    {
        "parseJSON": {}
    },
    {
        "typeConverter": {
            "entries": [
                {
                    "key": "status",
                    "type": "integer"
                }
            ]
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
    "name": "value",
    "status": 200
}
```

datetimeConverter

使用 datetimeConverter 处理器可以将日期时间字符串转换为您指定的格式。

字段	说明	必填？	默认	限制
source	对其应用日期转换的键。	是		最多条目数：10

字段	说明	必填？	默认	限制
matchPatterns	与 source 字段匹配的模式列表	是		最多条目数：5
target	用于存储结果的 JSON 字段。	是		最大长度：128 最大嵌套键深度：3
targetFormat	用于目标字段中的转换后数据的日期时间格式。	否	yyyy-MM-dd'T'HH:mm:ss.SSS'Z	最大长度：64
sourceTimezone	源字段的时区。 有关可能值的列表，请参阅 Java Supported Zone Ids and Offsets。	否	UTC	最小长度：1
目标时区	目标字段的时区。 有关可能值的列表，请参阅 Java Supported Zone Ids and Offsets。	否	UTC	最小长度：1
locale	源字段的区域设置。 有关可能值的列表，请参阅 Java 中的 Locale getAvailableLocales () 方法和示例。	是		最小长度：1

示例

以以下日志事件为例：

```
{"german_datetime": "Samstag 05. Dezember 1998 11:00:00"}
```

转换器配置如下，将 dateTimeConverter 和 parseJSON 结合使用：

```
[
```

```
    {
        "parseJSON": {}
    },
    {
        "dateTimeConverter": {
            "source": "german_datetime",
            "target": "target_1",
            "locale": "de",
            "matchPatterns": ["EEEE dd. MMMM yyyy HH:mm:ss"],
            "sourceTimezone": "Europe/Berlin",
            "targetTimezone": "America/New_York",
            "targetFormat": "yyyy-MM-dd'T'HH:mm:ss z"
        }
    }
]
```

转换后的日志事件将如下所示。

```
{
    "german_datetime": "Samstag 05. Dezember 1998 11:00:00",
    "target_1": "1998-12-05T17:00:00 MEZ"
}
```

转换指标和错误

CloudWatch 日志将转换指标发布到 CloudWatch。这些指标包括 `TransformedLogEvents`、`TransformedBytes` 和 `TransformationErrors`。有关更多信息，请参阅 日志转换器指标和维度。

每当 Lo CloudWatch gs 尝试转换日志事件但失败时，它都会向该日志事件添加一个`@transformationError`系统字段。运行 CloudWatch Logs Insights 查询时，您将在转换失败的所有日志事件中看到此字段。您可以使用一个查询来查询此字段，例如 `filter ispresent(@transformationError)` 可查找所有失败的转换事件。

使用 S3 表格集成访问日志

S3 表格与集成 CloudWatch 允许您访问 CloudWatch 使用分析引擎（例如亚马逊 Athena、Amazon Redshift 和支持 Apache 商店连接的第三方工具）获取的日志数据。 Iceberg-compatible 这种集成使您能够使用自己喜欢的工具执行全面的日志分析，并将 CloudWatch 日志中的数据与非CloudWatch数据相关联。此功能无需支付额外费用。

了解 S3 表集成

Amazon S3 表格集成是一种完全托管的解决方案，它使您在日志中的 CloudWatch 日志可以作为托管 Amazon S3 表使用。通过这种集成，除了日志功能外，您还可以更灵活地分析 CloudWatch 日志。

该集成的工作原理是创建托管 Amazon S3 表存储桶 (`aws-cloudwatch`)，并根据数据源名称和类型将特定的日志源与 Amazon S3 表关联起来（可通过日志控制台的 “日志管理” > “数据源” 选项卡进行管理）。 CloudWatch 关联后，即可通过 Apache Iceberg 格式的 Amazon S3 表访问 CloudWatch 日志数据。这种格式为各种分析引擎提供了一种高效查询数据的标准化方法。

核心组件

Apache Iceberg 表

S3 Tables 使用的基础表格式，可提供结构化数据存储并支持与多个分析引擎兼容。

数据源关联

根据数据源和类型标准将特定 CloudWatch 日志源链接到 S3 Tables 集成的过程。

数据流向 S3 表

了解 CloudWatch 日志和 S3 表之间的数据流动方式有助于您规划集成并有效地管理日志数据。

创建关联时，Lo CloudWatch gs 会自动将与关联数据源名称和类型匹配的新日志事件传送 CloudWatch-managed 到 S3 表存储桶。这些事件出现在该数据源的相应表下方的日志命名空间中。集成过程仅记录创建关联后收到的事件。现有日志数据不会回填。创建 S3 表集成时，如果您将 “启用所有日志源和类型在 S3 表中可用” 复选框保留为选中状态，则默认情况下，您的账户中的所有数据源都将自动关联并传输到 S3 表，包括将来添加的任何数据源。要仅向 S3 表提供特定的数据源，请在创建集成时清除此复选框，然后单独关联要包含的数据源。

S3 表存储桶中的数据保留与为日志组设置的保留策略相匹配。例如，如果您将日志组设置为 1 天的保留期，则 CloudWatch 日志将在一天后从 CloudWatch 日志和 S3 表中删除数据。当您删除日志组或日志流时，Lo CloudWatch gs 还会从 S3 表存储桶中删除数据。

何时使用 S3 表格集成

考虑使用 S3 表集成将日志数据与其他外部或非数据关联起来，或者当您更喜欢使用 Amazon Athena 等其他分析工具对日志CloudWatch 数据进行 CloudWatch 分析时。当您需要超出 CloudWatch 日志中可用范围的功能时，请使用此集成。在以下情况下，这种集成特别有价值：

- 您需要对大量日志数据运行复杂的 SQL-like 查询
- 您想将日志分析与现有分析工作流程和工具集成
- 您需要跨多个数据源的全面日志分析功能

除了现有的 CloudWatch 摄取和存储定价外，通过此集成创建的 S3 表无需支付额外的存储费用或表维护费用。

先决条件

在实施集成之前，请确保您具备以下条件：

- 现有 CloudWatch 日志数据
- 在 CloudWatch 日志和 S3 表之间进行跨服务访问的相应的 IAM 权限，如以下部分所述

IAM 权限

要将 CloudWatch 日志与 S3 表集成，您需要为两个不同的实体配置 IAM 权限：设置集成的用户或角色，以及 CloudWatch 日志承担的向 S3 表写入数据的服务角色。

适用于创建集成的角色或用户

设置集成的用户或角色需要以下权限：

- `observabilityadmin:CreateS3TableIntegration`创建集成并`logs:AssociateSourceToS3TableIntegration`添加源代码
- `s3tables:CreateTableBucket`、`s3tables:PutTableBucketEncryption`，以及`s3tables:PutTableBucketPolicy`配置 S3 表存储桶

对于服务角色

将以下 IAM 策略附加到 CloudWatch Logs 用来向表存储桶写入数据的 IAM 服务角色。此策略授予写入表的权限。将*aws-region123456789012*、和*log-group-name*，替换为您的 AWS 区域、账户 ID 和日志组名称。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:integrateWithS3Table"
            ],
            "Resource": ["arn:aws:logs:aws-region:123456789012:log-group:log-group-
name"],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

将以下信任策略附加到 Lo CloudWatch gs 将代入向 S3 表写入日志数据的 IAM 服务角色。您可以在集成设置期间创建或选择此角色。这些条件限制了该角色，因此 CloudWatch Logs 只能为指定的账户和日志组代入该角色。将*aws-region123456789012*、和*log-group-name*，替换为您的 AWS 区域、账户 ID 和日志组名称。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
```

```
                },
                "ArnLike": {
                    "aws:SourceArn": ["arn:aws:logs:aws-region:123456789012:log-
group:log-group-name"]
                }
            }
        }
    ]
}
```

KMS 密钥策略（适用于加密数据）

如果您使用客户托管密钥加密日志数据，则必须向 CloudWatch 服务主体和 S3 Tables 维护服务主体授予对该密钥的访问权限。将以下语句添加到您的 KMS 密钥策略中。将占位符值替换为您的 AWS 账户 ID、区域、KMS 密钥 ID 以及 S3 表或表存储桶 ARN。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnableSystemTablesKeyUsage",
            "Effect": "Allow",
            "Principal": {
                "Service": "systemtables.cloudwatch.amazonaws.com"
            },
            "Action": [
                "kms:DescribeKey",
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:aws-region:123456789012:key/key-id",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        },
        {
            "Sid": "EnableKeyUsage",
            "Effect": "Allow",
            "Principal": {
                "Service": "maintenance.s3tables.amazonaws.com"
            },
```

```
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:aws-region:123456789012:key/key-id",
            "Condition": {
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "<table-or-table-bucket-arn>/*"
                }
            }
        }
    ]
}
```

开始使用

要开始使用 S3 表集成，您需要在 CloudWatch 日志和 S3 表之间设置集成。此过程包括配置数据源关联和设置适当的 IAM 权限。

创建 S3 表格集成

1. 打开 CloudWatch 日志控制台，网址https://console.aws.amazon.com/cloudwatch/为”。
2. 选择设置、全局、创建 S3 表集成。
3. 自定义 S3 表中日志的加密方式，以及 CloudWatch 日志将用于将日志写入 S3 表的角色。
4. 如果您希望所有数据源自动与集成关联，请将 “启用所有日志源和类型在 S3 表中可用” 复选框保留为选中状态（默认情况下处于选中状态）。如果您只想关联特定的数据源，请清除此复选框。
5. 选择创建 S3 表集成。

> **Note**
>
> 如果您选择了在创建期间允许所有日志源和类型在 S3 表中可用，则会自动关联所有数据源，包括将来添加的任何数据源。你可以在这里停下来。仅当您清除该复选框并想要关联特定数据源时，才需要执行以下步骤。

将源关联到 S3 表集成

1. 打开 CloudWatch 日志控制台，网址https://console.aws.amazon.com/cloudwatch/为”。

2. 选择设置、全局、管理 S3 表集成。
3. 选择 “关联数据源”。
4. 选择要为其启用集成的数据源名称和数据源类型。
5. 选择 “关联数据源”。

从日志管理页面将源关联到 S3 表集成

1. 打开 CloudWatch 日志控制台，网址https://console.aws.amazon.com/cloudwatch/为”。
2. 在导航窗格中选择 “日志管理”。
3. 选择 “数据源” 选项卡。
4. 选择要集成的数据源名称和数据源类型。
5. 选择 “数据源操作”。
6. 选择与 S3 表集成 “关联”。
7. 查看数据源，然后选择 “关联数据源”。

在使用数据之前，您必须执行以下 3 个步骤：

1. 将亚马逊 S3 表格与 AWS 分析服务集成-使用亚马逊 S3 控制台
2. 配置 Lake Formation 权限
3. Connect 使用分析工具

将亚马逊 Amazon S3 表格与 AWS 分析服务-使用 Amazon S3 控制台 (Link)

使用 S3 控制台启用 S3 表集成

1. 打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择表存储桶。
3. 选择顶部的 “启用集成”。
4. 首次在任何区域集成表存储桶时，Amazon S3 会代表您创建一个新的 IAM 服务角色。此角色允许 Lake Formation 访问您账户中的所有表存储桶，并在 G AWS lue 数据目录中联合访问您的表。

配置 Lake Formation 权限

虽然 CloudWatch 日志有权写入表（在前面的步骤中配置），但用户和分析角色不会自动拥有读取数据的权限。您必须使用 AWS Lake Formation 明确授予访问权限。您必须为每位想要提供表访问权限的 IAM 委托人执行此操作。

向用户或角色授予查询权限

您必须向将在 Athena 或 Redshift 中运行查询的 IAM 委托人（用户或角色）授予选择和描述权限。

1. 打开 AWS Lake Formation 控制台。
2. 在导航窗格的权限下，选择数据湖权限。
3. 选择授权。
4. 委托人：选择需要访问权限的 IAM 用户或角色（例如，您的数据分析师或您当前正在使用的管理员角色）。
5. LF-Tags 或目录资源：选择命名数据目录资源。
6. 数据库和表：
   - 选择 CloudWatch 集成创建的 S3 表存储桶（`aws-cloudwatch`）。
   - 选择与您的数据源关联的特定表（可选）。
7. 表权限：选择 “选择并描述”。
8. 选择授权。

> **Note**
>
> 如果您在 Athena 中查询日志时遇到 “访问被拒绝” 错误，请确保运行查询的用户同时拥有适用于 Athena 的相应的 IAM 权限和上面定义的 Lake Formation 权限。

要详细了解 Lake Formation 权限，请https://docs.aws.amazon.com/lake-formation/latest/dg/granting-catalog-permissions.html访问。

Connect 使用分析工具

授予权限后，您可以将首选分析服务配置为查询 S3 表。S3 表格使用 Apache Iceberg 格式，亚马逊 Athena、Amazon Redshift 和 Amazon EMR 原生支持该格式。

在 Amazon Athena 中查询日志数据

Amazon Athena 通过 Amazon S3 表格目录与 S3 表进行交互。

设置 Athena 以查询您的日志数据

1. 打开亚马逊 Athena 主机，网址为。https://console.aws.amazon.com/athena/
2. 在查询编辑器中，从数据来源下拉列表中选择 Amazon S3 表类数据存储服务目录。
3. 如果您看不到目录，请确保您已完成上述针对特定用户角色的 Lake Formation 权限步骤。
4. 选择目录后，您的日志表将出现在数据库列表中。现在，您可以对日志数据运行标准 SQL 查询。

查询示例：`SELECT * FROM "amazon_vpc__flow" LIMIT 100;`

要详细了解如何将分析服务与 S3 表连接起来，请访问https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-tables-integrating-aws.html。

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 日志使用这些指标过滤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。

如果您已经配置 AWS Organizations 并正在使用成员帐户，则可以使用日志集中将日志数据从源账户收集到中央监控账户。

如果使用集中式日志组，则当创建指标筛选条件时，您可以使用以下系统字段维度：

- `@aws.account`-此维度代表日志事件源自的 AWS 账户 ID。
- `@aws.region`-此维度表示生成日志事件的 AWS 区域。

这些维度有助于识别日志数据的来源，从而对从集中式日志派生的指标进行更精细的筛选和分析。有关更多信息，请参阅 Cross-account 跨区域日志集中化。

如果具有订阅的日志组使用日志转换，则会将筛选模式应用于日志事件的转换版本。有关更多信息，请参阅 在摄取期间转换日志。

> **Note**
>
> 仅标准日志类中的日志组支持指标筛选条件。有关日志类的更多信息，请参阅 日志类。

在查看这些指标或设置警报时，您可以使用任何类型的 CloudWatch 统计数据，包括百分位统计数据。

> **Note**
>
> 仅当没有任何指标的值为负值时，才支持适用于指标的百分位数统计数据。如果您将指标筛选器设置为可以报告负数，则当它的值有负数时，该指标的百分位数统计数据不可用。有关更多信息，请参阅百分位数。

筛选条件不会以回溯方式筛选数据。筛选条件只会发布在创建后发生的事件的指标数据点。测试筛选模式时，筛选结果预览最多会显示前 50 条匹配的日志行以进行验证。如果筛选结果的时间戳早于指标创建时间，则不显示任何日志。

> **Note**
>
> 指标筛选器确保至少一次将日志事件传送到指定指标，而偶尔可能会发生重复传输。

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。由于维度是指标唯一标识符的一部分，因此每当从日志中提取唯一 name/value 对时，您就会创建该指标的新变体。

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

应将受监控日志信息发布到的 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**
>
> 指标筛选条件与 CloudWatch 日志见解查询有何不同
> 指标筛选器与 L CloudWatch ogs 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 日志见解语言查询语法。
> 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 日志指标。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 在您的日志事件中出现的次数，则可以指定名为 `ErrorCode` 的维度，以显示包含单词 ERROR 的日志事件总数，并按报告的错误代码筛选数据。

> **Tip**
>
> 为指标分配度量单位时，请确保指定正确的单位。如果稍后更改单位，则您的更改可能无法生效。有关 CloudWatch 支持的单位的完整列表，请参阅MetricDatum《亚马逊 CloudWatch API 参考》。

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，则在指标筛选条件未找到匹配项时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 防止汇总参差不齐的指标。 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 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 创建指标筛选器，使用由 JSON 和空格分隔的日志事件生成的指标发布维度。维度是 name/value 值对，仅适用于 JSON 和空格分隔的过滤器模式。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：

- 亚马逊 CloudWatch 用户指南中的尺寸
- 示例：从 Apache 日志中提取字段并在 Amazon L CloudWatch ogs 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。为了防止您意外收取高额费用，如果亚马逊在一定时间内针对指定维度生成 1000 个不同的 name/value 货币对，则可能会禁用您的指标筛选器。
> 您可以创建账单告警，通知您预估费用。有关更多信息，请参阅创建账单告警以监控 AWS 预估费用。

使用来自 JSON 日志事件的指标发布维度

以下示例包含描述如何在 JSON 指标筛选条件中指定维度的代码段。

Example: JSON log event

```
{
  "eventType": "UpdateTrail",
  "sourceIPAddress": "111.111.111.111",
  "arrayKey": [
        "value",
        "another value"
  ],
  "objectList": [
       {"name": "a",
         "id": 1
       },
       {"name": "b",
```

```
        "id": 2
      }
  ]

}
```

> **Note**
>
> 如果您使用示例 JSON 录入事件测试示例指标筛选条件，则必须在单行中输入示例 JSON 日志。

Example: Metric filter

每当 JSON 录入事件包含属性 `eventType` 和 `"sourceIPAddress"` 时，指标筛选条件都会增加指标。

```
{ $.eventType = "*" && $.sourceIPAddress != 123.123.* }
```

在创建 JSON 指标筛选条件时，您可以将指标筛选条件中的任何属性指定为维度。例如，要设置 `eventType` 为维度，请使用以下内容：

```
"eventType" : $.eventType
```

示例指标包含名为 `"eventType"` 的维度，并且示例录入事件中维度的值为 `"UpdateTrail"`。

使用来自以空格分隔的日志事件的指标发布维度

以下示例包含描述如何在以空格分隔的指标筛选条件中指定维度的代码段。

Example: Space-delimited log event

```
127.0.0.1 Prod frank [10/Oct/2000:13:25:15 -0700] "GET /index.html HTTP/1.0" 404
 1534
```

Example: Metric filter

```
[ip, server, username, timestamp, request, status_code, bytes > 1000]
```

当以空格分隔的录入事件包含筛选条件中指定的任何字段时，指标筛选条件将增加指标。例如，指标筛选条件在以空格分隔的录入事件示例中查找以下字段和值。

```
{
   "$bytes": "1534",
   "$status_code": "404",

   "$request": "GET /index.html HTTP/1.0",
   "$timestamp": "10/Oct/2000:13:25:15 -0700",
   "$username": "frank",
   "$server": "Prod",
   "$ip": "127.0.0.1"
}
```

在创建以空格分隔的指标筛选条件时，您可以将指标筛选条件中的任何字段指定为维度。例如，要设置 `server` 为维度，请使用以下内容：

```
"server" : $server
```

示例指标筛选条件包含名为 `server` 的维度，并且示例录入事件中维度的值为 `"Prod"`。

Example: Match terms with AND (&&) and OR (||)

您可以使用逻辑运算符 AND（"&&"）和 OR（"||"）创建包含条件的以空格分隔的指标筛选条件。以下指标筛选条件会返回日志事件，其中事件中的第一个单词是 ERROR 或 WARN 的任何超字符串。

```
[w1=ERROR || w1=%WARN%, w2]
```

使用录入事件中的值来增加指标的值

您可以创建指标筛选条件，该筛选条件会发布在录入事件中找到的数字值。本部分中的过程使用以下示例指标筛选条件来显示可以如何将 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. 在处打开 CloudWatch 控制台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 Logs 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 防止汇总参差不齐的指标。 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. 在处打开 CloudWatch 控制台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）如果您使用的是集中式日志组，则可以在筛选条件选择标准下，根据源账户（`@aws.account`）、源区域（`@aws.region`）或这两个条件指定筛选条件。
7. （可选）要测试您的筛选条件模式，请在 Test Pattern（测试模式）下，输入一个或多个用于测试模式的日志事件。每个日志事件必须格式化为一行。换行符用于分隔 Log event messages（日志事件消息）框中的日志事件。
8. 选择 Next（下一步），然后为指标筛选条件输入名称。
9. 在指标详细信息下的指标命名空间中，输入将发布指标的 CloudWatch 命名空间的名称。如果该命名空间尚不存在，请确保选中 Create new（新建）。
10. 对于 Metric name（指标名称），为新指标输入名称。

11. 对于 Metric value（指标值），如果您的指标筛选条件正在计算筛选条件中关键字的出现次数，请输入 1。对于包含其中一个关键字的每个日志事件，此操作会以 1 为增量增加该指标。

    或者，输入令牌，例如 **$size**。对于包含 size 字段的每个日志事件，此操作会以 size 字段中的数值为增量增加该指标。

12. （可选）对于 Unit（单位），选择要分配给相应指标的单位。如果未指定单位，则单位将设置为 None。

13. （可选）为指标输入多达三个维度的名称和令牌。如果将维度分配给指标筛选条件创建的指标，则无法为这些指标分配默认值。

    > **Note**
    >
    > 维度仅在 JSON 或以空格分隔的指标筛选条件中受支持。

14. 选择 Create metric filter（创建指标筛选条件）。可以从导航窗格找到您创建的指标筛选条件。选择 Logs（日志），然后选择 Log groups（日志组）。选择您为其创建指标筛选条件的日志组的名称，然后选择 Metric filters（指标筛选条件）选项卡。

示例：对日志事件进行计数

最简单的日志事件监控就是对发生的日志事件进行计数。您可能想对所有事件进行计数，以创建“检测信号”式监视器，或只是练习创建指标筛选条件。

在以下 CLI 示例中，名 MyAppAccessCount 为的指标筛选器应用于日志组 MyApp/access.log ，以在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. 在处打开 CloudWatch 控制台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。您应该看到发布到该指标的数据点 MyAppAccessEventCount。

要发布事件数据，请使用 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 对于每个包含错误的事件，日志会向MyApp/message.log命名空

间 ErrorCount 中的 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. 在处打开 CloudWatch 控制台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 Metric Details（指标详细信息）下，为 Metric Namespace（指标命名空间）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。

要发布活动，请使用 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，每个日志事件（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. 在处打开 CloudWatch 控制台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"
```

在放置这些示例日志事件后不久，您可以检索 CloudWatch 控制台中命名为的指标 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. 在处打开 CloudWatch 控制台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. 在处打开 CloudWatch 控制台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 调用中使用以下数据来测试此规则。如果不删除前例中的监控规则，则会生成两个不同的指标。

```
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /~test/ HTTP/1.1" 200 3
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:51:34 -0700] "GET /~test/index.html HTTP/1.1" 200 3
```

列出指标筛选条件

您可以列出日志组的所有指标筛选条件。

使用 CloudWatch 控制台列出指标筛选条件

1. 在处打开 CloudWatch 控制台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. 在处打开 CloudWatch 控制台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）。

要删除指标筛选器，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
```

Real-time 使用订阅处理日志数据

您可以使用订阅来访问来自 CloudWatch 日志的日志事件的实时提要，并将其传送到其他服务，例如 Amazon Kinesis 流、Amazon Data Firehose 流，或者 AWS Lambda 进行自定义处理、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。

您还可以使用 CloudWatch 日志集中化将日志数据从多个账户和区域复制到一个中心位置。有关更多信息，请参阅 Cross-account 跨区域日志集中化。

要开始订阅日志事件，请创建接收资源，例如 Amazon Kinesis Data Streams 流，事件将在其中传送。订阅过滤器定义过滤器模式，用于筛选哪些日志事件传送到您的 AWS 资源，以及有关将匹配的日志事件发送到何处的信息。日志事件在摄取后不久（通常不到三分钟）就会发送到接收资源。

> **Note**
>
> 如果具有订阅的日志组使用日志转换，则会将筛选模式与日志事件的转换版本进行比较。有关更多信息，请参阅 在摄取期间转换日志。

您可以在账户级别和日志组级别创建订阅。每个账户对于每个区域可以有一个账户级订阅筛选条件。每个日志组最多可以有五个与之关联的订阅过滤器。

> **Note**
>
> 如果目标服务返回可重试错误，例如限制异常或可重试服务异常（例如 HTTP 5xx），则 CloudWatch 日志最多会在 24 小时内继续重试交付。 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，订阅筛选器最多禁用 10 分钟，然后 L CloudWatch ogs 会重试将日志发送到目标。在此禁用期间，将跳过日志。

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

您还可以使用 CloudWatch 日志订阅将日志数据近乎实时地传输到亚马逊 OpenSearch 服务集群。有关更多信息，请参阅将 CloudWatch 日志数据流式传输到亚马逊 OpenSearch 服务。

仅标准日志类中的日志组支持订阅。有关日志类的更多信息，请参阅 日志类。

> **Note**
>
> 订阅筛选条件可能会批量记录事件以优化传输并减少对目的地的调用量。不保证使用批处理，但会尽可能地使用。

要按计划批处理和分析日志数据，请考虑使用使用计划查询自动进行日志分析。计划查询会自动运行 CloudWatch Logs Insights 查询，并将结果传送到 Amazon S3 存储桶或 Amazon EventBridge 事件总线等目的地。

> **Note**
>
> 订阅过滤器可确保至少一次传送事件，而重复的事件偶尔可能会发生。

内容

- 概念
- 日志组级别订阅筛选条件
- Account-level 订阅过滤器
- Cross-account 跨区域订阅
- 混淆代理问题防范
- 防止日志递归

概念

每个订阅筛选条件都由以下关键元素组成：

筛选条件模式

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亚马逊 Kinesis 数据流、Firehose 流或 Lambda 函数的亚马逊资源名称 (ARN)。

角色 ARN

一个 IAM 角色，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。

> **Note**
>
> 如果您的账户中启用了角色管理器，则会为您 CloudWatch 附加角色，此处描述的角色选项（例如“创建新角色”按钮）将替换为“自定义”选项。要使用其它角色，请选择自定义。有关创建 IAM 角色的更多信息，请参阅 IAM 用户指南中的 IAM 角色创建。

分配

当目标是 Amazon Kinesis Data Streams 流中的流时，用于将日志数据分配到目标的方法。默认情况下，日志数据按日志流进行分组。为了实现更均匀的分配，您可以对日志数据进行随机分组。

对于日志组级别的订阅，还包括下面的关键元素：

日志组名称

要将订阅筛选条件关联到的日志组。上传到此日志组的所有日志事件都受订阅筛选条件的约束，与该筛选条件匹配的日志事件将被传输到接收匹配日志事件的目标服务。

对于账户级别的订阅，还包括下面的关键元素：

选择标准

用于选择哪些日志组应用了账户级别订阅筛选条件的标准。如果不指定此元素，则账户级别订阅筛选条件将应用于账户中的所有日志组。此字段用于防止无限日志循环。有关无限日志循环问题的更多信息，请参阅日志 防止日志递归。

选择标准的大小限制为 25 KB。

对于集中式日志组，还包括下面的关键元素。这些元素可用作字段选择标准，以帮助识别日志数据的来源，从而对从集中式日志派生的指标进行更精细的筛选和分析。

@aws.account

此字段标识日志事件源自的 AWS 账户 ID。

@aws.region

此字段标识生成日志事件的 AWS 区域。

@source .log

此字段标识日志事件源自的源日志组。

日志组级别订阅筛选条件

您可以在亚马逊 Kinesis Data Streams AWS Lambda、Amazon Data Firehose 或亚马逊 OpenSearch 服务中使用订阅筛选条件。通过订阅筛选条件发送到服务的日志将经过 base64 编码并使用 gzip 格式进行压缩。如果您使用集中式日志 AWS Organizations，则可以选择发出`@aws.account`、`@aws.region`和`@source.log`系统字段，以确定哪些数据来自组织中的哪些账户、区域和源日志组。本部分提供了创建 CloudWatch 日志订阅筛选条件的示例，该过滤器将日志数据发送到 Firehose、Lambda、亚马逊 Kinesis 数据流和服务。 OpenSearch

> **Note**
>
> 如果要搜索日志数据，请参阅 Filter and pattern syntax。

示例

- 示例 1：使用 Amazon Kinesis 数据流的订阅筛选条件
- 示例 2：订阅筛选条件为 AWS Lambda
- 示例 3：Amazon Data Firehose 订阅筛选条件
- 示例 4：亚马逊 OpenSearch 服务的订阅筛选条件

示例 1：使用 Amazon Kinesis 数据流的订阅筛选条件

以下示例将订阅筛选条件与包含 AWS CloudTrail 事件的日志组关联。订阅筛选器将由 “根” AWS 凭证记录的每项活动传送到亚马逊 Kinesis 数据流中名为 “” 的流RootAccess。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或放random置订阅筛选条件创建订阅筛选条件distribution时指定。默认情况下，流筛选条件分配按日志流进行，这可能会导致节流。
> - 使用 CloudWatch 指标监控您的直播。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目标时限制 CloudWatch 日志的日志事件的数量。有关监控的更多信息，请参阅使用 CloudWatch 指标进行监控。
> - 在 Amazon Kinesis 数据流中对数据流使用按需容量模式。 On-demand模式可立即适应您的工作负载的增加或下降。有关按需容量模式的更多信息，请参阅 On-demand 模式。
> - 限制您的 CloudWatch 订阅筛选模式，使其与 Amazon Kinesis 数据流中的数据流容量相匹配。如果您向流发送的数据过多，则可能需要减小筛选器大小或调整筛选条件。

为 Amazon Kinesis Data Streams 创建订阅过滤器

1. 使用以下命令创建目标 流：

```
$ C:\>  aws kinesis create-stream --stream-name "RootAccess" --shard-count 1
```

2. 请耐心等待，直到 流变为活动状态（这可能需要一两分钟）。您可以使用以下 Amazon Kinesis Data Streams des cribe-stream 命令来检查该属性。 StreamDescription.StreamStatus 此外，请记下该StreamDescription.StreamARN值，因为在后面的步骤中将需要它：

```
aws kinesis describe-stream --stream-name "RootAccess"
```

下面是示例输出：

```
{
    "StreamDescription": {
        "StreamStatus": "ACTIVE",
        "StreamName": "RootAccess",
        "StreamARN": "arn:aws:kinesis:us-east-1:123456789012:stream/RootAccess",
```

```
        "Shards": [
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "EndingHashKey": "340282366920938463463374607431768211455",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
                    "49551135218688818456679503831981458784591352702181572610"
                }
            }
        ]
    }
}
```

3. 创建 IAM 角色，该角色将向 CloudWatch Logs 授予将数据放入数据流的权限。首先，您需要在文件（例如 ~/TrustPolicyForCWL-Kinesis.json）中创建信任策略。使用文本编辑器创建此策略。请勿使用 IAM 控制台来创建策略。

   此策略包括 aws:SourceArn 全局条件上下文密钥，有助于避免出现混淆代理安全问题。有关更多信息，请参阅 混淆代理问题防范。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
        "ArnLike": { "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。记下返回的Role.Arn值，因为在以后的步骤中也需要它：

```
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哪些操作。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 A ctive 状态并创建 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， CloudWatch 日志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。您可以通过获取亚马逊 Kinesis 数据流分片迭代器并使用 Amazon Kinesis 数据流获取记录命令来获取一些亚马逊 Kinesis 数据流记录，来验证是否发生了这种情况：

```
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，在 Amazon Kinesis 数据流开始返回数据之前，您可能需要多次拨打此电话。

您将看到包含一组记录的响应。亚马逊 Kinesis 数据流记录中的数据属性采用 base64 编码并使用 gzip 格式压缩。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 Amazon Kinesis 数据流记录，主要用于检查目的地是否可达。

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：订阅筛选条件为 AWS Lambda

在此示例中，您将创建一个 CloudWatch 日志订阅过滤器，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. 压缩文件 hello World.js 并使用名称保存`helloWorld.zip`。
4. 使用以下命令，其中的角色是您在第一步中设置的 Lambda 执行角色：

```
aws lambda create-function \
    --function-name helloworld \
    --zip-file fileb://file-path/helloWorld.zip \
    --role lambda-execution-role-arn \
    --handler helloWorld.handler \
    --runtime nodejs12.x
```

5. 授予 CloudWatch Logs 执行函数的权限。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。

   要查看您的 Lambda 函数的输出，请导航到 Lambda 函数，您将在//h aws/lambda elloworld 中看到输出：

   ```
   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：Amazon Data Firehose 订阅筛选条件

在此示例中，您将创建 CloudWatch 日志订阅，将与您定义的过滤条件相匹配的所有传入日志事件发送到您的 Amazon Data Firehose 传输流。从 CloudWatch 日志发送到亚马逊 Data Firehose 的数据已

经使用 gzip 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，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，请参阅将 CloudWatch 日志发送到 Firehose 。

> **Note**
>
> 在创建 Firehose 流之前，请计算将生成的日志数据量。请确保创建一个处理此数据量的 Firehose 流。如果流无法处理卷，则日志流将受限制。有关 Firehose 流数据量限制的更多信息，请参阅 Amazon Data Firehose Data Limits。

为 Firehose 创建订阅筛选条件

1. 创建 Amazon Simple Storage Service（Amazon S3）存储桶。我们建议您使用专为 CloudWatch 日志创建的存储桶。但是，如果要使用现有存储桶，请跳至第 2 步。

   运行以下命令，将占位符区域替换为您想使用的区域：

   ```
   aws s3api create-bucket --bucket amzn-s3-demo-bucket2 --create-bucket-configuration
    LocationConstraint=region
   ```

   下面是示例输出：

   ```
   {
       "Location": "/amzn-s3-demo-bucket2"
   }
   ```

2. 创建 IAM 角色，该角色将授予 Amazon Data Firehose 将数据放入您的 Amazon S3 存储桶的权限。

   有关更多信息，请参阅《Amazon Data Firehose 开发人员指南》中的 Controlling Access with Amazon Data Firehose。

   首先，使用文本编辑器在文件 ~/TrustPolicyForFirehose.json 中创建一个信任策略，具体如下所示：

   ```
   {
     "Statement": {
       "Effect": "Allow",
       "Principal": { "Service": "firehose.amazonaws.com" },
       "Action": "sts:AssumeRole"
   ```

```
    }
}
```

3. 使用 create-role 命令创建 IAM 角色，并指定信任策略文件。记下返回Role.Arn值，因为在后面的步骤中会用到它：

```
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 可对您的账户执行的操作。首先，使用文本编辑器在文件 ~/PermissionsForFirehose.json 中创建权限策略：

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "s3:AbortMultipartUpload",
          "s3:GetBucketLocation",
          "s3:GetObject",
          "s3:ListBucket",
          "s3:ListBucketMultipartUploads",
          "s3:PutObject" ],
```

```
      "Resource": [
          "arn:aws:s3:::amzn-s3-demo-bucket2",
          "arn:aws:s3:::amzn-s3-demo-bucket2/*" ]
    }
  ]
}
```

5. 使用以下 put-role-policy 命令将权限策略与角色关联：

```
aws iam put-role-policy --role-name FirehosetoS3Role --policy-name Permissions-
Policy-For-Firehose --policy-document file://~/PermissionsForFirehose.json
```

6. 如下所示，创建目标 Firehose 传输流，将 RoleARN 和 BucketARN 的占位符值替换为您创建的角色和存储桶 ARN：

```
aws firehose create-delivery-stream \
   --delivery-stream-name 'my-delivery-stream' \
   --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::123456789012:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::amzn-s3-demo-bucket2"}'
```

请注意，Firehose 会自动为已交付的 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. 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 Firehose describe-delivery-stream 命令来检查属性。 DeliveryStreamDescription.DeliveryStreamStatus 此外，请记下该DeliveryStreamDescription.DeliveryStreamARN值，因为在后面的步骤中将需要它：

```
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:::amzn-s3-demo-bucket2",
                    "BufferingHints": {
                        "IntervalInSeconds": 300,
                        "SizeInMBs": 5
                    }
                }
            }
        ]
    }
}
```

8. 创建 IAM 角色，向 CloudWatch 日志授予将数据放入您的 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": {
         "ArnLike": {
             "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Role.Arn值，因为在后面的步骤中会用到它：

```
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哪些操作。首先，使用文本编辑器创建权限策略文件（例如 ~/PermissionsForCWL.json）：

```
{
    "Statement":[
      {
        "Effect":"Allow",
        "Action":["firehose:PutRecord"],
        "Resource":[
            "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-
name"]
      }
    ]
}
```

11. 使用 put-role-policy 命令将权限策略与角色关联：

```
aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-
name Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json
```

12. 在 Amazon Data Firehose 传输流处于活动状态并创建 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启动从所选日志组到您的 Amazon Data Firehose 传输流的实时日志数据流动：

```
aws logs put-subscription-filter \
    --log-group-name "CloudTrail" \
    --filter-name "Destination" \
    --filter-pattern "{$.userIdentity.type = Root}" \
    --destination-arn "arn:aws:firehose:region:123456789012:deliverystream/my-
delivery-stream" \
    --role-arn "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
```

13. 设置订阅筛选条件后，CloudWatch 日志会将所有与筛选模式相匹配的传入日志事件转发到您的 Amazon Data Firehose 传输流。根据 Amazon Data Firehose 传输流上设置的时间缓冲间隔，数据将开始显示在 Amazon S3 中。经过足够的时间后，您可以通过检查您的 Amazon S3 存储桶验证您的数据。

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket2' --prefix 'firehose/'
{
    "Contents": [
        {
            "LastModified": "2015-10-29T00:01:25.000Z",
            "ETag": "\"a14589f8897f4089d3264d9e2d1f1610\"",
            "StorageClass": "STANDARD",
            "Key": "firehose/2015/10/29/00/my-delivery-stream-2015-10-29-00-01-21-
a188030a-62d2-49e6-b7c2-b11f1a7ba250",
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b5"
            },
            "Size": 593
        },
        {
            "LastModified": "2015-10-29T00:35:41.000Z",
            "ETag": "\"a7035b65872bb2161388ffb63dd1aec5\"",
            "StorageClass": "STANDARD",
            "Key": "firehose/2015/10/29/00/my-delivery-
stream-2015-10-29-00-35-40-7cc92023-7e66-49bc-9fd4-fc9819cc8ed3",
```

```
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b6"
            },
            "Size": 5752
        }
    ]
}
```

```
aws s3api get-object --bucket 'amzn-s3-demo-bucket2' --key 'firehose/2015/10/29/00/
my-delivery-stream-2015-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250'
 testfile.gz

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

Amazon S3 对象中的数据以 gzip 格式压缩。您可以使用以下 Unix 命令检查命令行中的原始数据：

```
zcat testfile.gz
```

示例 4：亚马逊 OpenSearch 服务的订阅筛选条件

在此示例中，您将创建 CloudWatch 日志订阅，将与您定义的过滤器相匹配的传入日志事件发送到您的 OpenSearch 服务域。

为 OpenSearch 服务创建订阅过滤器

1. 创建 OpenSearch 服务域。有关更多信息，请参阅创建 OpenSearch 服务域
2. 在处打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/。
3. 在导航窗格中，选择日志组。
4. 选择日志组的名称。
5. 选择操作、订阅筛选条件、创建亚马逊 OpenSearch 服务订阅筛选条件。

6. 选择是要流式传输到此账户还是其他账户中的集群。
   - 如果选择此账户，请选择您在步骤 1 中创建的域。
   - 如果选择其他账户，请输入该域的 ARN 和端点。
7. 如果您选择了其他账户，请提供域 ARN 和端点。
8. 对于 Amazon S OpenSearch ervice 集群，选择将传送日志组数据的集群的名称
9. 选择一种日志格式。
10. 对于订阅筛选条件模式，输入要在您的日志事件中查找的字词或模式。这样可以确保您仅将感兴趣的数据发送到您的 OpenSearch 服务集群。有关更多信息，请参阅 指标筛选条件的筛选条件模式语法。
11. （可选）对于 Select log data to test（选择要测试的日志数据），请选择一个日志流，然后选择 Test pattern（测试模式），以确认搜索筛选器是否会返回您期望的结果。
12. 选择 Start streaming（开始流式传输）。

Account-level 订阅过滤器

> **Important**
>
> 订阅筛选条件可能会出现无限递归循环的风险，如果不加以解决，可能会导致提取计费大幅增加。为了降低这种风险，我们建议您在账户级订阅筛选条件中使用选择标准来排除从属于订阅交付工作流一部分的资源中提取日志数据的日志组。有关此问题以及确定要排除哪些日志组的更多信息，请参阅 防止日志递归。

您可以设置一个账户级别的订阅策略，其包括账户中的一部分日志组。账户订阅政策适用于亚马逊 Kinesis Data Streams 或 Amazon Data Firehose。 AWS Lambda通过账户级订阅策略发送到服务的日志采用 base64 编码并使用 gzip 格式进行压缩。本节提供了一些示例，您可以按照示例为亚马逊 Kinesis 数据流、Lambda 和 Firehose 创建账户级订阅。

> **Note**
>
> 要查看您账户中所有订阅筛选策略的列表，请使用 `describe-account-policies` 命令并将 `--policy-type` 参数的值设置为 `SUBSCRIPTION_FILTER_POLICY`。有关更多信息，请参阅 describe-account-policies¶。

示例

- 示例 1：使用 Amazon Kinesis 数据流的订阅筛选条件
- 示例 2：订阅筛选条件为 AWS Lambda
- 示例 3：Amazon Data Firehose 订阅筛选条件

示例 1：使用 Amazon Kinesis 数据流的订阅筛选条件

在创建用于账户级订阅政策的 Amazon Kinesis Data Streams 数据流之前，请计算将生成的日志数据量。请确保创建具有足够分片的 流来处理此卷。如果某个流没有足够的分片，则会受到限制。有关流量限制的更多信息，请参阅 Amazon Kinesis 数据流文档中的配额和限制。

> **Warning**
>
> 由于多个日志组的日志事件被转发到目的地，因此存在节流的风险。受限的交付项最多可在 24 小时内重试。24 小时后，失败的交付项将被丢弃。
> 要减轻节流的风险，您可以执行以下步骤：
>
> - 使用 CloudWatch 指标监控您的亚马逊 Kinesis 数据流。这可以帮助您识别节流并相应地调整配置。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。
> - 在 Amazon Kinesis 数据流中对数据流使用按需容量模式。 On-demand模式可立即适应您的工作负载的增加或下降。有关更多信息，请参见 On-demand 模式。
> - 限制您的 CloudWatch 日志订阅筛选模式，使其与 Amazon Kinesis 数据流中的数据流容量相匹配。如果您向流发送的数据过多，则可能需要减小筛选器大小或调整筛选条件。

以下示例使用账户级订阅策略将所有日志事件转发到 Amazon Kinesis 数据流中的一个流。筛选模式将所有日志事件与文本进行匹配，`Test`并将其转发到 Amazon Kinesis 数据流中的流。

为 Amazon 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. 等待几分钟让流变为活动状态。您可以使用 describe-stream 命令检查该属性来验证该StreamDescription.StreamStatus直播是否处于活动状态。

```
aws kinesis describe-stream --stream-name "TestStream"
```

下面是示例输出：

```
{
    "StreamDescription": {
        "StreamStatus": "ACTIVE",
        "StreamName": "TestStream",
        "StreamARN": "arn:aws:kinesis:region:123456789012:stream/TestStream",
        "Shards": [
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "EndingHashKey": "EXAMPLE84634633746074317682114555",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
                    "EXAMPLE688818456679503831981458784591352702181572610"
                }
            }
        ]
    }
}
```

3. 创建 IAM 角色，该角色将向 CloudWatch Logs 授予将数据放入数据流的权限。首先，您需要在文件（例如 ~/TrustPolicyForCWL-Kinesis.json）中创建信任策略。使用文本编辑器创建此策略。

此策略包括 aws:SourceArn 全局条件上下文密钥，有助于避免出现混淆代理安全问题。有关更多信息，请参阅 混淆代理问题防范。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
        "ArnLike": { "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。记下返回的Role.Arn值，因为在以后的步骤中也需要它：

```
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哪些操作。首先，您将在文件（例如 ~/PermissionsForCWL-Kinesis.json）中创建权限策略。使用文本编辑器创建此策略。不要使用 IAM 控制台来创建它。

```
{
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:123456789012:stream/TestStream"
    }
  ]
}
```

6. 使用以下 put-role-policy 命令将权限策略与角色关联：

```
aws iam put-role-policy  --role-name CWLtoKinesisRole  --policy-name Permissions-Policy-For-CWL  --policy-document file://~/PermissionsForCWL-Kinesis.json
```

7. 在直播处于 A ctive 状态并创建 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。在此示例中，所有包含字符串 ERROR 的日志事件都将流式传输，但名为 LogGroupToExclude1 和 LogGroupToExclude2 的日志组中的日志事件除外。

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

8. 设置订阅过滤器后， CloudWatch 日志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您的直播。

   selection-criteria 字段是可选的，但对于从订阅筛选条件中排除可能导致无限日志递归的日志组非常重要。有关此问题以及确定要排除哪些日志组的更多信息，请参阅 防止日志递归。目前，NOT IN 是 selection-criteria 唯一支持的运算符。

   您可以通过使用 Amazon Kinesis Data Streams 分片迭代器并使用 Amazon Kinesis Data Streams get-records 命令获取一些 Amazon Kinesis 数据流记录来验证日志事件的流向：

```
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 Amazon Kinesis 数据流开始返回数据之前，您可能需要多次使用此命令。

您将看到包含一组记录的响应。亚马逊 Kinesis 数据流记录中的数据属性采用 base64 编码并使用 gzip 格式压缩。您可使用以下 Unix 命令检查命令行中的原始数据：

```
echo -n "<Content of Data>" | base64 -d | zcat
```

Base64 解码和解压缩数据被格式化为 JSON 并具有以下结构：

```
{
    "messageType": "DATA_MESSAGE",
    "owner": "123456789012",
    "logGroup": "Example1",
    "logStream": "logStream1",
    "subscriptionFilters": [
        "ExamplePolicy"
    ],
    "logEvents": [
        {
            "id": "31953106606966983378809025079804211143289615424298221568",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221569",
            "timestamp": 1432826855000,
```

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

数据架构中的关键元素如下：

messageType

数据消息将使用“DATA_MESSAGE”类型。有时， CloudWatch 日志可能会发出“CONTROL_MESSAGE” 类型的 Amazon Kinesis 数据流记录，主要用于检查目的地是否可达。

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。

policyLevel

强制实施策略的级别。“ACCOUNT_LEVEL_POLICY”是账户级订阅筛选策略的 `policyLevel`。

示例 2：订阅筛选条件为 AWS Lambda

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量，则日志流将受限制。由于所有日志组或账户的部分日志组的日志事件都被转发到目标，因此存在节流风险。有关 Lambda 限制的更多信息，请参阅 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. 压缩文件 hello World.js 并使用名称保存`helloWorld.zip`。

4. 使用以下命令，其中的角色是您在第一步中设置的 Lambda 执行角色：

```
aws lambda create-function \
    --function-name helloworld \
```

```
    --zip-file fileb://file-path/helloWorld.zip \
    --role lambda-execution-role-arn \
    --handler helloWorld.handler \
    --runtime nodejs18.x
```

5. 授予 CloudWatch Logs 执行函数的权限。使用以下命令，将占位符账户替换为您自己的账户。

```
aws lambda add-permission \
    --function-name "helloworld" \
    --statement-id "helloworld" \
    --principal "logs.amazonaws.com" \
    --action "lambda:InvokeFunction" \
    --source-arn "arn:aws:logs:region:123456789012:log-group:*" \
    --source-account "123456789012"
```

6. 使用以下命令创建账户级订阅筛选策略，将占位符账户替换为您自己的账户。在此示例中，所有包含字符串 ERROR 的日志事件都将流式传输，但名为 LogGroupToExclude1 和 LogGroupToExclude2 的日志组中的日志事件除外。

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

设置订阅过滤器后， CloudWatch 日志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您的直播。

selection-criteria 字段是可选的，但对于从订阅筛选条件中排除可能导致无限日志递归的日志组非常重要。有关此问题以及确定要排除哪些日志组的更多信息，请参阅 防止日志递归。目前，NOT IN 是 selection-criteria 唯一支持的运算符。

7. （可选）使用样本日志事件进行测试。在出现命令提示符时，运行以下命令，以将一条简单的日志消息放入已订阅的流中。

要查看您的 Lambda 函数的输出，请导航到 Lambda 函数，您将在//h aws/lambda elloworld 中看到输出：

```
aws logs put-log-events --log-group-name Example1 --log-stream-name logStream1 --
log-events "[{\"timestamp\":CURRENT TIMESTAMP MILLIS , \"message\": \"Simple Lambda
 Test\"}]"
```

您将看到包含一组 Lambda 的响应。Lambda 记录中的数据属性采用 base64 编码并使用 gzip 格式进行压缩。Lambda 接收的实际负载采用以下格式：`{ "awslogs": {"data": "BASE64ENCODED_GZIP_COMPRESSED_DATA"} }`。您可以使用以下 Unix 命令检查命令行中的原始数据：

```
echo -n "<BASE64ENCODED_GZIP_COMPRESSED_DATA>" | base64 -d | zcat
```

Base64 解码和解压缩数据被格式化为 JSON 并具有以下结构：

```
{
    "messageType": "DATA_MESSAGE",
    "owner": "123456789012",
    "logGroup": "Example1",
    "logStream": "logStream1",
    "subscriptionFilters": [
        "ExamplePolicyLambda"
    ],
    "logEvents": [
        {
            "id": "31953106606966983378809025079804211143289615424298221568",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221569",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221570",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":
\"Root\"}"
        }
```

```
    ],
    "policyLevel": "ACCOUNT_LEVEL_POLICY"
}
```

> **Note**
>
> 账户级订阅筛选条件将不会应用于目标 Lambda 函数的日志组。这是为了防止可能导致提取计费增加的无限日志递归。有关此问题的更多信息，请参阅 防止日志递归。

数据架构中的关键元素如下：

messageType

数据消息将使用“DATA_MESSAGE”类型。有时， CloudWatch 日志可能会发出“CONTROL_MESSAGE” 类型的 Amazon Kinesis 数据流记录，主要用于检查目的地是否可达。

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。

policyLevel

强制实施策略的级别。“ACCOUNT_LEVEL_POLICY”是账户级订阅筛选策略的 `policyLevel`。

示例 3：Amazon Data Firehose 订阅筛选条件

在此示例中，您将创建 CloudWatch 日志账户级别订阅筛选器策略，将与您定义的过滤条件相匹配的传入日志事件发送到您的 Amazon Data Firehose 传送流。从 CloudWatch 日志发送到亚马逊 Data Firehose 的数据已经使用 gzip 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，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 CloudWatch 日志写入 Kinesis Data Firehose。

> **Warning**
>
> 在创建 Firehose 流之前，请计算将生成的日志数据量。请确保创建一个处理此数据量的 Firehose 流。如果流无法处理卷，则日志流将受限制。有关 Firehose 流数据量限制的更多信息，请参阅 Amazon Data Firehose Data Limits。

为 Firehose 创建订阅筛选条件

1. 创建 Amazon Simple Storage Service（Amazon S3）存储桶。我们建议您使用专为 CloudWatch 日志创建的存储桶。但是，如果要使用现有存储桶，请跳至第 2 步。

   运行以下命令，将占位符区域替换为您想使用的区域：

   ```
   aws s3api create-bucket --bucket amzn-s3-demo-bucket2 --create-bucket-configuration
    LocationConstraint=region
   ```

   下面是示例输出：

   ```
   {
       "Location": "/amzn-s3-demo-bucket2"
   }
   ```

2. 创建 IAM 角色，该角色将授予 Amazon Data Firehose 将数据放入您的 Amazon S3 存储桶的权限。

   有关更多信息，请参阅《Amazon Data Firehose 开发人员指南》中的 Controlling Access with Amazon Data Firehose。

   首先，使用文本编辑器在文件 ~/TrustPolicyForFirehose.json 中创建一个信任策略，具体如下所示：

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "firehose.amazonaws.com" },
    "Action": "sts:AssumeRole"
    }
}
```

3. 使用 create-role 命令创建 IAM 角色，并指定信任策略文件。记下返回Role.Arn值，因为在后面的步骤中将需要它：

```
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 可对您的账户执行的操作。首先，使用文本编辑器在文件 ~/PermissionsForFirehose.json 中创建权限策略：

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
            "s3:AbortMultipartUpload",
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:ListBucketMultipartUploads",
            "s3:PutObject" ],
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket2",
            "arn:aws:s3:::amzn-s3-demo-bucket2/*" ]
      }
    ]
}
```

5. 使用以下 put-role-policy 命令将权限策略与角色关联：

```
aws iam put-role-policy --role-name FirehosetoS3Role --policy-name Permissions-
Policy-For-Firehose --policy-document file://~/PermissionsForFirehose.json
```

6. 如下所示，创建目标 Firehose 传输流，将 RoleARN 和 BucketARN 的占位符值替换为您创建的角色和存储桶 ARN：

```
aws firehose create-delivery-stream \
   --delivery-stream-name 'my-delivery-stream' \
   --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::123456789012:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::amzn-s3-demo-bucket2"}'
```

nFireHose 会自动为已交付的 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. 等待几分钟让流变为活动状态。你可以使用 Firehose describe- delivery-stream 命令来检查属性。 DeliveryStreamDescription.DeliveryStreamStatus 此外，请记下该DeliveryStreamDescription.DeliveryStreamARN值，因为在后面的步骤中将需要它：

```
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:::amzn-s3-demo-bucket2",
                    "BufferingHints": {
                        "IntervalInSeconds": 300,
                        "SizeInMBs": 5
                    }
                }
            }
        ]
    }
}
```

8. 创建 IAM 角色，向 CloudWatch 日志授予将数据放入您的 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": {
         "ArnLike": {
             "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Role.Arn值，因为在后面的步骤中将需要它：

```
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哪些操作。首先，使用文本编辑器创建权限策略文件（例如 ~/PermissionsForCWL.json）：

```
{
    "Statement":[
      {
        "Effect":"Allow",
        "Action":["firehose:PutRecord"],
        "Resource":[
            "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-
name"]
      }
```

```
    ]
}
```

11. 使用 put-role-policy 命令将权限策略与角色关联：

```
aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-
name Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json
```

12. 在 Amazon Data Firehose 传输流处于活动状态并创建 IAM 角色后，您可以创建 CloudWatch 日志账户级别的订阅筛选策略。该策略将立即启动从所选日志组到您的 Amazon Data Firehose 传输流的实时日志数据流动：

```
aws logs put-account-policy \
    --policy-name "ExamplePolicyFirehose" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document '{"RoleArn":"arn:aws:iam::123456789012:role/
CWLtoKinesisFirehoseRole", "DestinationArn":"arn:aws:firehose:us-
east-1:123456789012:deliverystream/delivery-stream-name", "FilterPattern": "Test",
 "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

13. 设置订阅筛选条件后，CloudWatch 日志会将与筛选模式相匹配的传入日志事件转发到您的 Amazon Data Firehose 传输流。

    `selection-criteria` 字段是可选的，但对于从订阅筛选条件中排除可能导致无限日志递归的日志组非常重要。有关此问题以及确定要排除哪些日志组的更多信息，请参阅 防止日志递归。目前，NOT IN 是 `selection-criteria` 唯一支持的运算符。

    根据 Amazon Data Firehose 传输流上设置的时间缓冲间隔，数据将开始显示在 Amazon S3 中。经过足够的时间后，您可以通过检查您的 Amazon S3 存储桶验证您的数据。

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket2' --prefix 'firehose/'
{
    "Contents": [
        {
            "LastModified": "2023-10-29T00:01:25.000Z",
            "ETag": "\"a14589f8897f4089d3264d9e2d1f1610\"",
            "StorageClass": "STANDARD",
            "Key": "firehose/2015/10/29/00/my-delivery-stream-2015-10-29-00-01-21-
a188030a-62d2-49e6-b7c2-b11f1a7ba250",
```

```
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b5"
            },
            "Size": 593
        },
        {
            "LastModified": "2015-10-29T00:35:41.000Z",
            "ETag": "\"a7035b65872bb2161388ffb63dd1aec5\"",
            "StorageClass": "STANDARD",
            "Key": "firehose/2023/10/29/00/my-delivery-stream-2023-10-29-00-35-40-
EXAMPLE-7e66-49bc-9fd4-fc9819cc8ed3",
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "EXAMPLE6be062b19584e0b7d84ecc19237f87b6"
            },
            "Size": 5752
        }
    ]
}
```

```
aws s3api get-object --bucket 'amzn-s3-demo-bucket2' --key 'firehose/2023/10/29/00/
my-delivery-stream-2023-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250'
 testfile.gz

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

Amazon S3 对象中的数据以 gzip 格式压缩。您可以使用以下 Unix 命令检查命令行中的原始数据：

```
zcat testfile.gz
```

Cross-account 跨区域订阅

您可以与其他 AWS 账户的所有者合作，接收他们在您的 AWS 资源上的日志事件，例如 Amazon Kinesis 或 Amazon Data Firehose 数据流（这称为跨账户数据共享）。例如，可以从集中的 Amazon Kinesis 数据流或 Firehose 流中读取此日志事件数据，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实时事件数据流并将其交付给信息安全组，他们可以使用 Amazon Kinesis Data Streams 将数据附加到其现有的安全分析系统。

> **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 Region-specific 资源都是在美国东部（弗吉尼亚北部）创建的。

如果您已配置 AWS Organizations 并正在使用成员帐户，则可以使用日志集中将日志数据从源账户收集到中央监控账户。

如果使用集中式日志组，则当创建订阅筛选条件时，您可以使用以下系统字段维度：

- `@aws.account`-此维度代表日志事件源自的 AWS 账户 ID。
- `@aws.region`-此维度表示生成日志事件的 AWS 区域。
- `@source.log`-此维度代表日志事件源自的源日志组。

这些维度有助于识别日志数据的来源，从而对从集中式日志派生的指标进行更精细的筛选和分析。

主题

- Cross-account 使用 Amazon Kinesis 数据流进行跨区域日志数据共享
- Cross-account 使用 Firehose 进行跨区域日志数据共享
- Cross-account 使用 Amazon Kinesis 数据流进行跨区域账户级订阅
- Cross-account 使用 Firehose 进行跨区域账户级别订阅

Cross-account 使用 Amazon Kinesis 数据流进行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，并告知 L CloudWatch ogs 已准备好将其日志事件发送到指定目的地。在本节其余部分的过程中，将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 — 设置一个封装 Amazon Kinesis 数据流的目的地，并告知 CloudWatch 日志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。在本节其余部分的过程中，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

您要用作订阅源目标的 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，向 CloudWatch Logs 授予将数据放入所选流所需的权限。

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 资源可以位于不同的区域。在以下部分的示例中，所有 Region-specific 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：跨账户目标的 Add/validate 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。

在此示例中，日志数据接收者账户的 AWS 账户 ID 为 999999999999，而日志数据发送者 AWS 账户 ID 为 111111111111。

此示例使用名为的 Amazon Kinesis 数据流创建了一个目的地 RecipientStream，以及一个允许 CloudWatch 日志向其写入数据的角色。

创建目标后，L CloudWatch ogs 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 会代表源账户向目标发送 CloudWatch 日志事件。

创建目标

1. 在收款人账户中，在 Amazon Kinesis 数据流中创建目标流。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 命令来检查该属性。StreamDescription.StreamStatus 此外，请记下该StreamDescription.StreamARN值，因为稍后会将其传递给 CloudWatch Logs：

```
aws kinesis describe-stream --stream-name "RecipientStream"
{
  "StreamDescription": {
    "StreamStatus": "ACTIVE",
    "StreamName": "RecipientStream",
    "StreamARN": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "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，向 CloudWatch Logs 授予将数据放入数据流的权限。首先，你需要在文件 ~/TrustPolicyForCWL.json 中创建信任策略。使用文本编辑器创建此策略文件，请勿使用 IAM 控制台来创建。

   此策略包括指定 `sourceAccountId` 的 `aws:SourceArn` 全局条件上下文密钥，有助于避免出现混淆代理安全问题。如果您在第一次调用中还不知道源账户 ID，则建议您将目标 ARN 放在源 ARN 字段中。在随后的调用中，应将源 ARN 设置为从第一次调用中收集的实际源 ARN。有关更多信息，请参阅 混淆代理问题防范。

```
{
    "Statement": {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.amazonaws.com"
        },
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:region:sourceAccountId:*",
                    "arn:aws:logs:region:recipientAccountId:*"
                ]
            }
        },
        "Action": "sts:AssumeRole"
    }
}
```

4. 使用 aws iam create-role 命令创建 IAM 角色，并指定信任策略文件。记下返回 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中创建权限策略 ~/ PermissionsForCWL.json:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:999999999999:stream/RecipientStream"
    }
  ]
}
```

6. 使用 aws iam put-role-policy 命令将权限策略与角色关联：

```
aws iam put-role-policy \
    --role-name CWLtoKinesisRole \
    --policy-name Permissions-Policy-For-CWL \
    --policy-document file://~/PermissionsForCWL.json
```

7. 在直播处于活动状态并创建 IAM 角色后，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DestinationArn负载中返回的：

```
aws logs put-destination \
    --destination-name "testDestination" \
    --target-arn "arn:aws:kinesis:region:999999999999:stream/RecipientStream" \
    --role-arn "arn:aws:iam::999999999999:role/CWLtoKinesisRole"

{
  "DestinationName" : "testDestination",
  "RoleArn" : "arn:aws:iam::999999999999:role/CWLtoKinesisRole",
```

```
  "DestinationArn" : "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
  "TargetArn" : "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream"
}
```

b. 在步骤 7a 完成后，在日志数据接收者账户中将访问策略与目标关联。此策略必须指定 logs:PutSubscriptionFilter 操作并向发件人账户授予访问目标的权限。

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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "logs:PutSubscriptionFilter",
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": [
                        "o-1234567890"
                    ]
                }
            }
        }
    ]
}
```

下一个示例只允许日志数据发件人账户（111111111111）将日志发送到日志数据收件人账户。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111111111111"
            },
            "Action": "logs:PutSubscriptionFilter",
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination"
        }
    ]
}
```

c. 将您在上一步中创建的策略附加到目标。

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

此访问策略允许 ID 为 111111111111 的 AWS 账户中的用户使用 ARN: aws: logs:: 999999999999: destination: testDestination: testDestination: testDestination 向目的地拨PutSubscriptionFilter打电话。*region*任何其他用户尝试拨 PutSubscriptionFilter打该目的地的呼叫都将被拒绝。

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 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 可以假设并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器。

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。

要创建跨账户日志订阅所需的 IAM 角色，请使用 AWS Organizations

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 L CloudWatch ogs 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。

第 3 步：跨账户目标的 Add/validate 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 直播），您必须在发送账户中拥有基于身份的策略，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 Cross-account 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"
}
```

记下由*RoleName*或表示的值*UserName*。

2. 登录发送账户，使用您在步骤 1 AWS 管理控制台 中输入的命令的输出中返回的 IAM 角色或 IAM 用户搜索附加的策略。
3. 验证附加到该角色或用户的策略是否提供在跨账户目标资源上调用 `logs:PutSubscriptionFilter` 的明确权限。

   以下策略仅提供在单个 AWS 账户、账户中对任何目标资源创建订阅筛选器的权限999999999999：

   JSON

```
{
     "Version":"2012-10-17",
     "Statement": [
        {
            "Sid": "AllowSubscriptionFiltersOnAccountResources",
            "Effect": "Allow",
            "Action": "logs:PutSubscriptionFilter",
            "Resource": [
                "arn:aws:logs:*:*:log-group:*",
                "arn:aws:logs:*:123456789012:destination:*"
            ]
        }
    ]
}
```

   以下策略仅提供在单个 AWS 账户、账户sampleDestination中命名的特定目标资源上创建订阅筛选器的权限123456789012：

   JSON

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

```
            "Resource": [
                "arn:aws:logs:*:*:log-group:*",
                "arn:aws:logs:*:123456789012:destination:sampleDestination"
            ]
        }
    ]
}
```

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 凭据记录的每项活动都将传送到您之前创建的目标。该目标封装了一个名为 “RecipientStream” 的流。

以下部分中的其余步骤假设您已按照AWS CloudTrail 用户指南中将 CloudTrail 事件发送到 CloudWatch 日志中的说明进行操作，并创建了包含您的 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 第 3 步：跨账户目标的 Add/validate 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，目的地可以指向位于不同区域的 AWS 资源，例如 Amazon Kinesis 数据流。

验证日志事件流

创建订阅筛选器后，CloudWatch 日志会将与过滤器模式匹配的所有传入日志事件转发到封装在名为 “” RecipientStream 的目标流中的流。目标所有者可以通过使用 a ws kinesis get-shard-iterator 命令获取亚马逊 Kinesis 数据流分片，然后使用 a ws kinesis get- records 命令获取一些亚马逊 Kinesis 数据流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 Amazon Kinesis 数据流开始返回数据之前，您可能需要重新运行几次 get-records 命令。

您应该看到包含一系列亚马逊 Kinesis 数据流记录的响应。亚马逊 Kinesis 数据流记录中的数据属性以 gzip 格式压缩，然后进行 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 Amazon Kinesis 数据流记录，主要用于检查目的地是否可达。

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 TestD estination 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：

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "222222222222"
            },
            "Action": "logs:PutSubscriptionFilter",
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination"
        }
    ]
}
```

3. 调PutDestinationPolicy用将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 启用从的日志记录 AWS 务 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是否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 L CloudWatch ogs 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。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "logs:PutSubscriptionFilter",
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": [
                        "o-1234567890"
                    ]
                }
            }
        }
    ]
}
```

2. 输入以下命令，将刚创建的策略附加到现有目标。要更新目标以使用带有组织 ID 的访问策略，而不是列出特定 AWS 账户 ID 的访问策略，请添加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从的日志记录 AWS 务，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过滤器，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
```

Cross-account 使用 Firehose 进行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，并告知 L CloudWatch ogs 已准备好将其日志事件发送到指定目的地。在本节其余部分的过程中，将显示日志数据发送者的虚构 AWS 账号为 111111111111。
- 日志数据接收者 — 设置一个封装 Amazon Kinesis 数据流的目的地，并告知 CloudWatch 日志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。在本节其余部分的过程中，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 存储的 Firehose 传输流。您还可以使用不同的设置来设置 Firehose 传输流。有关更多信息，请参阅 Creating a Firehose Delivery Stream。

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

> **Note**
>
> 支持同一账户和跨区域传输流的 Firehose 订阅筛选条件。

主题

- 步骤 1：创建 Firehose 传输流
- 步骤 2：创建目标
- 第 3 步：跨账户目标的 Add/validate IAM 权限
- 步骤 4：创建订阅筛选条件
- 验证录入事件流
- 在运行时修改目标成员资格

步骤 1：创建 Firehose 传输流

> **Important**
>
> 在完成以下步骤之前，您必须使用访问策略，以便 Firehose 可以访问您的 Amazon S3 存储桶。有关更多信息，请参阅《Amazon Data Firehose 开发人员指南》中的 Controlling Access。
> 本部分（步骤 1）中的所有步骤都需要在日志数据接收者账户中完成。
> 以下示例命令中使用的是美国东部（弗吉尼亚州北部）。请将此区域替换为适用于您的部署的正确区域。

创建将用作目标的 Firehose 传输流

1. 创建 Amazon S3 存储桶：

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --create-bucket-configuration
 LocationConstraint=us-east-1
```

2. 创建 IAM 角色，授予 Firehose 将数据放入存储桶的权限。

   a. 首先，使用文本编辑器在文件 ~/TrustPolicyForFirehose.json 中创建信任策略。

```
{ "Statement": { "Effect": "Allow", "Principal": { "Service":
 "firehose.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition":
 { "StringEquals": { "sts:ExternalId":"222222222222" } } } }
```

   b. 创建 IAM 角色，并指定您刚创建的信任策略文件。

```
aws iam create-role \
    --role-name FirehosetoS3Role \
    --assume-role-policy-document file://~/TrustPolicyForFirehose.json
```

   c. 此命令的输出与以下内容类似。记下角色名称和角色 ARN。

```
{
    "Role": {
        "Path": "/",
        "RoleName": "FirehosetoS3Role",
        "RoleId": "AROAR3BXASEKW7K635M53",
        "Arn": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
```

```
        "CreateDate": "2021-02-02T07:53:10+00:00",
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Effect": "Allow",
                "Principal": {
                    "Service": "firehose.amazonaws.com"
                },
                "Action": "sts:AssumeRole",
                "Condition": {
                    "StringEquals": {
                        "sts:ExternalId": "222222222222"
                    }
                }
            }
        }
    }
}
```

3. 创建权限策略，以定义 Firehose 可对您的账户执行的操作。

   a. 首先，使用文本编辑器在名为 ~/PermissionsForFirehose.json 的文件中创建以下权限策略。根据应用场景，您可能需要为此文件添加更多权限。

```
{
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
    }]
}
```

   b. 输入以下命令，以将刚创建的权限策略与 IAM 角色相关联。

```
aws iam put-role-policy --role-name FirehosetoS3Role --policy-name
 Permissions-Policy-For-Firehose-To-S3 --policy-document file://~/
PermissionsForFirehose.json
```

4. 输入以下命令以创建 Firehose 传输流。将*my-role-arn*和*amzn-s3-demo-bucket2-arn*替换为适合您的部署的正确值。

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

该输出值应该类似于以下内容：

```
{
    "DeliveryStreamARN": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream"
}
```

步骤 2：创建目标

> **Important**
>
> 此过程中的所有步骤都需要在日志数据接收者账户中完成。

创建目标后，L CloudWatch ogs 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. 请等待，直至您在 步骤 1：创建 Firehose 传输流 中创建的 Firehose 流变为活动状态。您可以使用以下命令来检查该StreamDescription.StreamStatus属性。

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

此外，请记下该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:::amzn-s3-demo-bucket",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    }
                },
                "ExtendedS3DestinationDescription": {
                    "RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
                    "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    },
                    "S3BackupMode": "Disabled"
                }
            }
```

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

您的传输流可能需要一两分钟才会显示为活动状态。

2. 当交付流处于活动状态时，创建 IAM 角色，该角色将向 L CloudWatch ogs 授予将数据放入您的 Firehose 流的权限。首先，你需要在文件 ~/ TrustPolicyForCWL.json 中创建信任策略。使用文本编辑器创建此策略。有关 CloudWatch 日志终端节点的更多信息，请参阅 Amazon L CloudWatch ogs 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": {
            "ArnLike": {
                "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": {
                    "ArnLike": {
                        "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中创建权限策略 ~/ PermissionsForCWL.json:

```
{
    "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创建 IAM 角色后，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。你可以使用文本编辑器将此策略放到 ~/ AccessPolicy.json 文件中：

      JSON

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

```
        "AWS" : "111111111111"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}
```

c. 这将创建一个策略，该策略定义了对目标具有写入权限的人。此策略必须指定 logs: PutSubscriptionFilter 操作才能访问目标。 Cross-account 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
```

第 3 步：跨账户目标的 Add/validate 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 直播），您必须在发送账户中拥有基于身份的策略，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 Cross-account 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"
}
```

记下由*RoleName*或表示的值*UserName*。

2. 登录发送账户，使用您在步骤 1 AWS 管理控制台 中输入的命令的输出中返回的 IAM 角色或 IAM 用户搜索附加的策略。
3. 验证附加到该角色或用户的策略是否提供在跨账户目标资源上调用 `logs:PutSubscriptionFilter` 的明确权限。

   以下策略仅提供在单个 AWS 账户、账户中对任何目标资源创建订阅筛选器的权限999999999999：

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid":
 "AllowSubscriptionFiltersOnAnyResourceInOneSpecificAccount",
            "Effect": "Allow",
            "Action": "logs:PutSubscriptionFilter",
            "Resource": [
                "arn:aws:logs:*:*:log-group:*",
                "arn:aws:logs:*:123456789012:destination:*"
            ]
        }
    ]
}
```

   以下策略仅提供在单个 AWS 账户、账户`sampleDestination`中命名的特定目标资源上创建订阅筛选器的权限123456789012：

   JSON

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

```
            "Action": "logs:PutSubscriptionFilter",
            "Resource": [
                "arn:aws:logs:*:*:log-group:*",
                "arn:aws:logs:*:123456789012:destination:amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

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 “根” AWS 凭据记录的每项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 第 3 步：跨账户目标的 Add/validate 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 数据流。

验证录入事件流

创建订阅筛选器后， CloudWatch 日志会将与过滤器模式匹配的所有传入日志事件转发到 Firehose 传送流。根据 Firehose 传输流上设置的时间缓冲间隔，数据将开始显示在您的 Amazon S3 存储桶中。经过足够的时间后，您可以通过检查 Amazon S3 存储桶来验证您的数据。要检查存储桶，请输入以下命令：

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket'
```

该命令的输出将与以下内容类似：

```
{
    "Contents": [
        {
            "Key": "2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba",
            "LastModified": "2021-02-02T09:00:26+00:00",
            "ETag": "\"EXAMPLEa817fb88fc770b81c8f990d\"",
            "Size": 198,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "firehose+2test",
                "ID": "EXAMPLE27fd05889c665d2636218451970ef79400e3d2aecca3adb1930042e0"
            }
        }
    ]
}
```

然后，您可以通过输入以下命令，从存储桶中检索特定对象。将 key 的值替换为您在上一个命令中找到的值。

```
aws s3api get-object --bucket 'amzn-s3-demo-bucket' --key '2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba' testfile.gz
```

Amazon S3 对象中的数据以 gzip 格式压缩。您可使用以下命令之一检查命令行中的原始数据：

Linux：

```
zcat testfile.gz
```

macOS：

```
zcat <testfile.gz
```

在运行时修改目标成员资格

您可能会遇到必须在您拥有的目标中添加或删除日志发送者的情况。您可以通过新的访问策略在目的地上使用该PutDestinationPolicy操作。在以下示例中，将阻止之前添加的账户 111111111111 再发送任何日志数据，并将启用账户 333333333333。

1. 获取当前与目标 TestD estination 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：

   JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "333333333333 "
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
```

```
}
```

3. 使用以下命令将NewAccessPolicy.json文件中定义的策略与目标相关联：

```
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。

Cross-account 使用 Amazon Kinesis 数据流进行跨区域账户级订阅

创建跨账户订阅时，可以指定单个账户或企业作为发件人。如果指定企业，则此过程允许企业中的所有账户向接收方账户发送日志。

要跨账户共享日志数据，您需要建立日志数据发送者和接收者：

- 日志数据发送者 —从收件人那里获取目标信息，并告知 L CloudWatch ogs 已准备好将其日志事件发送到指定目的地。在本节其余部分的过程中，将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 — 设置一个封装 Amazon Kinesis 数据流的目的地，并告知 CloudWatch 日志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。在本节其余部分的过程中，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

您要用作订阅源目标的 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，向 CloudWatch Logs 授予将数据放入所选流所需的权限。

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 资源可以位于不同的区域。在以下部分的示例中，所有 Region-specific 资源都是在美国东部（弗吉尼亚北部）创建的。

主题

- 设置新的跨账户订阅
- 更新现有的跨账户订阅

设置新的跨账户订阅

按照这些部分中所述步骤设置新的跨账户日志订阅。

主题

- 步骤 1：创建目标
- 步骤 2：（仅在使用企业时）创建 IAM 角色
- 步骤 3：创建账户级订阅筛选策略
- 验证日志事件流
- 在运行时修改目标成员资格

步骤 1：创建目标

> **Important**
>
> 此过程中的所有步骤都需要在日志数据接收者账户中完成。

在此示例中，日志数据接收者账户的 AWS 账户 ID 为 999999999999，而日志数据发送者 AWS 账户 ID 为 111111111111。

此示例使用名为的 Amazon Kinesis 数据流创建了一个目的地 RecipientStream，以及一个允许 CloudWatch 日志向其写入数据的角色。

创建目标后，L CloudWatch ogs 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. 在收款人账户中，在 Amazon Kinesis 数据流中创建目标流。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 命令来检查该属性。StreamDescription.StreamStatus 此外，请记下该StreamDescription.StreamARN值，因为稍后会将其传递给 CloudWatch Logs：

```
aws kinesis describe-stream --stream-name "RecipientStream"
{
  "StreamDescription": {
    "StreamStatus": "ACTIVE",
    "StreamName": "RecipientStream",
    "StreamARN": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "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，向 CloudWatch Logs 授予将数据放入数据流的权限。首先，你需要在文件 ~/TrustPolicyForCWL.json 中创建信任策略。使用文本编辑器创建此策略文件，请勿使用 IAM 控制台来创建。

   此策略包括指定 sourceAccountId 的 aws:SourceArn 全局条件上下文密钥，有助于避免出现混淆代理安全问题。如果您在第一次调用中还不知道源账户 ID，则建议您将目标 ARN 放在源 ARN 字段中。在随后的调用中，应将源 ARN 设置为从第一次调用中收集的实际源 ARN。有关更多信息，请参阅 混淆代理问题防范。

   ```
   {
       "Statement": {
           "Effect": "Allow",
           "Principal": {
               "Service": "logs.amazonaws.com"
           },
           "Condition": {
               "ArnLike": {
                   "aws:SourceArn": [
                       "arn:aws:logs:region:sourceAccountId:*",
                       "arn:aws:logs:region:recipientAccountId:*"
                   ]
               }
           },
           "Action": "sts:AssumeRole"
       }
   }
   ```

4. 使用 aws iam create-role 命令创建 IAM 角色，并指定信任策略文件。记下返回 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中创建权限策略 ~/ PermissionsForCWL.json:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:999999999999:stream/RecipientStream"
    }
  ]
}
```

6. 使用 aws iam put-role-policy 命令将权限策略与角色关联：

```
aws iam put-role-policy \
    --role-name CWLtoKinesisRole \
    --policy-name Permissions-Policy-For-CWL \
    --policy-document file://~/PermissionsForCWL.json
```

7. 在直播处于活动状态并创建 IAM 角色后，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DestinationArn负载中返回的：

```
aws logs put-destination \
    --destination-name "testDestination" \
    --target-arn "arn:aws:kinesis:region:999999999999:stream/RecipientStream" \
    --role-arn "arn:aws:iam::999999999999:role/CWLtoKinesisRole"

{
  "DestinationName" : "testDestination",
  "RoleArn" : "arn:aws:iam::999999999999:role/CWLtoKinesisRole",
  "DestinationArn" : "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
  "TargetArn" : "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream"
}
```

b. 在步骤 7a 完成后，在日志数据接收者账户中将访问策略与目标关联。此策略必须指定 logs:PutSubscriptionFilter 操作并向发件人账户授予访问目标的权限。

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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "logs:PutSubscriptionFilter",
                "logs:PutAccountPolicy"
            ],
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
            "Condition": {
                "StringEquals": {
```

```
                    "aws:PrincipalOrgID": [
                        "o-1234567890"
                    ]
                }
            }
        }
    ]
}
```

下一个示例只允许日志数据发件人账户（111111111111）将日志发送到日志数据收件人账户。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111111111111"
            },
            "Action": [
                "logs:PutSubscriptionFilter",
                "logs:PutAccountPolicy"
            ],
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination"
        }
    ]
}
```

c. 将您在上一步中创建的策略附加到目标。

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

此访问策略允许 ID 为 111111111111 的 AWS 账户中的用户使用 ARN: aws: logs:: 99999999999999: destination: testDestination: testDestination: testDestination 向目的地拨PutSubscriptionFilter打电话。*region*任何其他用户尝试拨 PutSubscriptionFilter打该目的地的呼叫都将被拒绝。

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 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 3：创建账户级订阅筛选策略。

步骤 2：（仅在使用企业时）创建 IAM 角色

在上一节中，如果您使用向账户 111111111111 所在的企业授予权限的访问策略创建了目标，而不是直接向账户 111111111111 授予权限，则请按照本节中的步骤操作。否则，您可以跳至 步骤 3：创建账户级订阅筛选策略。

本节中的步骤创建一个 IAM 角色，该角色 CloudWatch 可以假设并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器。

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。

要创建跨账户日志订阅所需的 IAM 角色，请使用 AWS Organizations

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 L CloudWatch ogs 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 3：创建账户级订阅筛选策略。

步骤 3：创建账户级订阅筛选策略

在创建目标后，日志数据接收者账户可与其他 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一个账户级订阅筛选条件。该筛选条件与发送者账户 111111111111 相关联，以便将与筛选条件和选择条件匹配的每个日志事件都传输到前面创建的目标。该目标封装了一个名为 “RecipientStream” 的流。

`selection-criteria` 字段是可选的，但对于从订阅筛选条件中排除可能导致无限日志递归的日志组非常重要。有关此问题以及确定要排除哪些日志组的更多信息，请参阅 防止日志递归。目前，NOT IN 是 `selection-criteria` 唯一支持的运算符。

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

发送者账户的日志组和目标必须位于同一 AWS 区域内。但是，目的地可以指向位于不同区域的 AWS 资源，例如 Amazon Kinesis 数据流。

验证日志事件流

创建账户级订阅筛选策略后， CloudWatch 日志会将所有与过滤模式和选择条件相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 a ws kinesis get-shard-iterator 命令获取亚马逊 Kinesis 数据流分片，然后使用 a ws kinesis get- records 命令获取一些亚马逊 Kinesis 数据流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 Amazon Kinesis 数据流开始返回数据之前，您可能需要重新运行该`get-records`命令几次。

您应该看到包含一系列亚马逊 Kinesis 数据流记录的响应。亚马逊 Kinesis 数据流记录中的数据属性以 gzip 格式压缩，然后进行 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
\"}"
        }
    ]
}
```

数据架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 Amazon Kinesis 数据流记录，主要用于检查目的地是否可达。

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。

policyLevel

强制实施策略的级别。“ACCOUNT_LEVEL_POLICY”是账户级订阅筛选策略的 `policyLevel`。

在运行时修改目标成员资格

您可能遇到必须在您拥有的目标中添加或删除某些用户的成员资格的情况。您可通过新访问策略对您的目标使用 `put-destination-policy` 命令。在以下示例中，将阻止之前添加的账户 111111111111 再发送任何日志数据，并将启用账户 222222222222。

1. 获取当前与目标 TestD estination 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：

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "222222222222"
            },
            "Action": [
                "logs:PutSubscriptionFilter",
                "logs:PutAccountPolicy"
            ],
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination"
        }
```

```
    ]
}
```

3. 调PutDestinationPolicy用将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 启用从的日志记录 AWS 务 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是否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 L CloudWatch ogs 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策略。

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

步骤 2：更新现有的目标访问策略

更新所有发件人账户中的订阅筛选条件后，可以更新收件人账户中的目标访问策略。

在以下示例中，收件人账户为 999999999999，目的地名为 testDestination。

此更新将使企业中 ID 为 o-1234567890 的所有账户向收件人账户发送日志。只有创建了订阅筛选条件的账户才会实际向收件人账户发送日志。

要更新收件人账户中的目标访问策略以开始使用企业 ID 获取权限

1. 在收件人账户中，请使用文本编辑器创建包含下列内容的 ~/AccessPolicy.json 文件。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "logs:PutSubscriptionFilter",
                "logs:PutAccountPolicy"
            ],
            "Resource": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": [
                        "o-1234567890"
                    ]
                }
            }
        }
    ]
}
```

2. 输入以下命令，将刚创建的策略附加到现有目标。要更新目标以使用带有组织 ID 的访问策略，而不是列出特定 AWS 账户 ID 的访问策略，请添加`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从的日志记录 AWS 务，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过滤器，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
```

Cross-account 使用 Firehose 进行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，并告知 L CloudWatch ogs 已准备好将其日志事件发送到指定目的地。在本节其余部分的过程中，将显示日志数据发送者的虚构 AWS 账号为 111111111111。
- 日志数据接收者 — 设置一个封装 Amazon Kinesis 数据流的目的地，并告知 CloudWatch 日志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。在本节其余部分的过程中，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 存储的 Firehose 传输流。您还可以使用不同的设置来设置 Firehose 传输流。有关更多信息，请参阅 Creating a Firehose Delivery Stream。

> **Note**
>
> 日志组和目标必须在同一个 AWS 区域中。但是，目标指向的 AWS 资源可以位于不同的区域。

> **Note**
>
> 支持同一账户和跨区域传输流的 Firehose 订阅筛选条件。

主题

- 步骤 1：创建 Firehose 传输流
- 步骤 2：创建目标
- 步骤 3：创建账户级订阅筛选策略
- 验证录入事件流
- 在运行时修改目标成员资格

步骤 1：创建 Firehose 传输流

> **Important**
>
> 在完成以下步骤之前，您必须使用访问策略，以便 Firehose 可以访问您的 Amazon S3 存储桶。有关更多信息，请参阅《Amazon Data Firehose 开发人员指南》中的 Controlling Access。
> 本部分（步骤 1）中的所有步骤都需要在日志数据接收者账户中完成。
> 以下示例命令中使用的是美国东部（弗吉尼亚州北部）。请将此区域替换为适用于您的部署的正确区域。

创建将用作目标的 Firehose 传输流

1. 创建 Amazon S3 存储桶：

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --create-bucket-configuration
 LocationConstraint=us-east-1
```

2. 创建 IAM 角色，授予 Firehose 将数据放入存储桶的权限。

   a. 首先，使用文本编辑器在文件 ~/TrustPolicyForFirehose.json 中创建信任策略。

```
{ "Statement": { "Effect": "Allow", "Principal": { "Service":
 "firehose.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition":
 { "StringEquals": { "sts:ExternalId":"222222222222" } } } }
```

   b. 创建 IAM 角色，并指定您刚创建的信任策略文件。

```
aws iam create-role \
    --role-name FirehosetoS3Role \
    --assume-role-policy-document file://~/TrustPolicyForFirehose.json
```

   c. 此命令的输出与以下内容类似。记下角色名称和角色 ARN。

```
{
    "Role": {
        "Path": "/",
        "RoleName": "FirehosetoS3Role",
        "RoleId": "AROAR3BXASEKW7K635M53",
        "Arn": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
```

```
        "CreateDate": "2021-02-02T07:53:10+00:00",
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Effect": "Allow",
                "Principal": {
                    "Service": "firehose.amazonaws.com"
                },
                "Action": "sts:AssumeRole",
                "Condition": {
                    "StringEquals": {
                        "sts:ExternalId": "222222222222"
                    }
                }
            }
        }
    }
}
```

3. 创建权限策略，以定义 Firehose 可对您的账户执行的操作。

   a. 首先，使用文本编辑器在名为 ~/PermissionsForFirehose.json 的文件中创建以下权限策略。根据应用场景，您可能需要为此文件添加更多权限。

```
{
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
    }]
}
```

   b. 输入以下命令，以将刚创建的权限策略与 IAM 角色相关联。

```
aws iam put-role-policy --role-name FirehosetoS3Role --policy-name
 Permissions-Policy-For-Firehose-To-S3 --policy-document file://~/
PermissionsForFirehose.json
```

4. 输入以下命令以创建 Firehose 传输流。将*my-role-arn*和*amzn-s3-demo-bucket2-arn*替换为适合您的部署的正确值。

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

该输出值应该类似于以下内容：

```
{
    "DeliveryStreamARN": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream"
}
```

步骤 2：创建目标

> **Important**
>
> 此过程中的所有步骤都需要在日志数据接收者账户中完成。

创建目标后，L CloudWatch ogs 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. 请等待，直至您在 步骤 1：创建 Firehose 传输流 中创建的 Firehose 流变为活动状态。您可以使用以下命令来检查该StreamDescription.StreamStatus属性。

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

此外，请记下该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:::amzn-s3-demo-bucket",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    }
                },
                "ExtendedS3DestinationDescription": {
                    "RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
                    "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    },
                    "S3BackupMode": "Disabled"
                }
            }
```

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

您的传输流可能需要一两分钟才会显示为活动状态。

2. 当交付流处于活动状态时，创建 IAM 角色，该角色将向 L CloudWatch ogs 授予将数据放入您的 Firehose 流的权限。首先，你需要在文件 ~/ TrustPolicyForCWL.json 中创建信任策略。使用文本编辑器创建此策略。有关 CloudWatch 日志终端节点的更多信息，请参阅 Amazon L CloudWatch ogs 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": {
            "ArnLike": {
                "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": {
                    "ArnLike": {
                        "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中创建权限策略 ~/ PermissionsForCWL.json:

```
{
    "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创建 IAM 角色后，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在 `~/AccessPolicy.json` 文件中：

      JSON

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

```
        "AWS" : "111111111111"
      },
      "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
      "Resource" : "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}
```

c. 这将创建一个策略，该策略定义了对目标具有写入权限的人。此策略必须指定logs:PutSubscriptionFilter和logs:PutAccountPolicy操作才能访问目标。Cross-account 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
```

步骤 3：创建账户级订阅筛选策略

切换到发送账户，在此示例中为 111111111111。您现在将在发送账户中创建账户级订阅筛选策略。在此示例中，筛选条件会导致除两个日志组之外的所有日志组中包含字符串 ERROR 的每个日志事件都传输到前面创建的目标位置。

```
aws logs put-account-policy \
    --policy-name "CrossAccountFirehoseExamplePolicy" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document '{"DestinationArn":"arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination", "FilterPattern":
 "{$.userIdentity.type = AssumedRole}", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

发送账户的日志组和目标必须位于同一 AWS 区域内。但是，目标可以指向位于不同区域的 AWS 资源，例如 Firehose 数据流。

验证录入事件流

创建订阅筛选器后，CloudWatch 日志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 传送流。根据 Firehose 传输流上设置的时间缓冲间隔，数据将开始显示在您的 Amazon S3 存储桶中。经过足够的时间后，您可以通过检查 Amazon S3 存储桶来验证您的数据。要检查存储桶，请输入以下命令：

```
aws s3api list-objects --bucket 'amzn-s3-demo-bucket'
```

该命令的输出将与以下内容类似：

```
{
    "Contents": [
        {
            "Key": "2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba",
            "LastModified": "2023-02-02T09:00:26+00:00",
            "ETag": "\"EXAMPLEa817fb88fc770b81c8f990d\"",
            "Size": 198,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "firehose+2test",
                "ID": "EXAMPLE27fd05889c665d2636218451970ef79400e3d2aecca3adb1930042e0"
            }
        }
    ]
}
```

然后，您可以通过输入以下命令，从存储桶中检索特定对象。将 key 的值替换为您在上一个命令中找到的值。

```
aws s3api get-object --bucket 'amzn-s3-demo-bucket' --key '2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba' testfile.gz
```

Amazon S3 对象中的数据以 gzip 格式压缩。您可使用以下命令之一检查命令行中的原始数据：

Linux：

```
zcat testfile.gz
```

macOS：

```
zcat <testfile.gz
```

在运行时修改目标成员资格

您可能会遇到必须在您拥有的目标中添加或删除日志发送者的情况。您可以通过新的访问策略在目的地上使用PutDestinationPolicy和PutAccountPolicy操作。在以下示例中，将阻止之前添加的账户 111111111111 再发送任何日志数据，并将启用账户 333333333333。

1. 获取当前与目标 TestD estination 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：

   JSON

   ```
   {
   ```

```
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "333333333333 "
      },
      "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
      "Resource" : "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}
```

3. 使用以下命令将NewAccessPolicy.json文件中定义的策略与目标相关联：

```
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，即不具有某操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在中 AWS，跨服务模仿可能会导致副手混乱的问题。 Cross-service 当一项服务（呼叫服务）调用另一项服务（被叫服务）时，可能会发生模拟。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。为防止这种情况，请 AWS 提供工具来帮助您保护所有服务的数据，这些服务主体已被授予访问您账户中资源的权限。

我们建议在资源策略中使用 aws:SourceArn、aws:SourceAccount、aws:SourceOrgID 和 aws:SourceOrgPaths 全局条件上下文键，以限制为其他服务提供的资源访问权限。使用 aws:SourceArn 来仅将一个资源与跨服务访问相关联。使用 aws:SourceAccount 来让该账户中的任何资源与跨服务使用相关联。使用 aws:SourceOrgID 来允许某组织内的任何账户中的任何资源与跨服务使用相关联。使用 aws:SourceOrgPaths 来将 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。

为授予 CloudWatch 日志的访问权限而记录的政策将数据写入到 Amazon Kinesis Data Streams 和 Firehose 中，步骤 1：创建目标并步骤 2：创建目标展示了如何使用aws:SourceArn全局条件上下文密钥来帮助防止出现混乱的副手问题。

防止日志递归

如果不加以阻止，订阅过滤器存在导致无限日志递归的风险，这可能会导致 CloudWatch 日志和目标中的摄取账单大幅增加。当订阅筛选条件与由于订阅传输工作流而接收日志事件的日志组关联时，会发生这种情况。提取到日志组中的日志将传输到目标位置，从而导致日志组提取更多日志，然后这些日志将再次转发到目标位置，从而形成递归循环。

例如，考虑一个目标位置为 Firehose 的订阅筛选条件，它将日志事件传输到 Amazon S3。此外，还有一个 Lambda 函数，可以处理传输到 Amazon S3 的新事件并自行生成一些日志。如果将订阅筛选条件应用于 Lambda 函数的日志组，则该函数生成的日志事件将转发到目标位置的 Firehose 和 Amazon S3，然后它们将再次调用该函数，从而生成更多日志并将其转发到 Firehose 和 Amazon S3，导致再次调用该函数等。这将陷入无限循环，导致日志提取、Firehose 和 Amazon S3 的计费意外增加。

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

我们建议您不要将订阅筛选条件应用于属于订阅传输工作流一部分的日志组。对于账户级订阅筛选条件，请使用 PutAccountPolicy API 中的 selectionCriteria 参数从策略中排除这些日志组。

排除日志组时，请考虑以下生成日志并可能成为订阅交付工作流程一部分的 AWS 服务：

- 带有 Fargate 的 Amazon EC2
- Lambda

- AWS 步进函数
- 启用日志的 Amazon VPC 流 CloudWatch 日志

> **Note**
>
> Lambda 目标的日志组生成的日志事件将不会被转发回账户级订阅筛选策略的 Lambda 函数。在这种情况下，账户订阅策略不需要使用 `selectionCriteria` 排除目标 Lambda 函数的日志组。

指标筛选条件、订阅筛选条件、筛选日志事件和 Live Tail 的筛选条件模式语法

> **Note**
>
> 有关如何使用 Amazon Logs Insights 查询语言查询 CloudWatch 日志组的信息，请参阅CloudWatch 日志见解语言查询语法。

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。

筛选模式构成了指标筛选条件、订阅筛选条件、日志事件和 Live Tail 用来匹配日志事件中的字词的语法。字词可以是单词、准确的短语或数字值。正则表达式（regex）可用于创建独立的筛选条件模式，也可以与 JSON 和以空格分隔的筛选条件模式合并。

使用要匹配的字词创建筛选条件模式。筛选条件模式仅返回包含您定义的字词的录入事件。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。

Multi-byte 不支持字符。

> **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。
- ?：匹配前面字词的零个或多个出现次数。例如，%colou?r% 可以同时匹配“color”和“colour”。
- []：定义字符类。匹配方括号内包含的字符列表或字符范围。例如，%[abc]% 匹配“a”、“b”或“c”；%[a-z]% 匹配从“a”到“z”的任何小写字母；%[abcx-z]% 匹配“a”、“b”、“c”、“x”、“y”或“z”。
- {m, n}：与前面的字词匹配至少 m 次且不超过 n 次。例如，%a{3,5}% 仅匹配“aaa”、“aaaa”和“aaaaa”。

  > **Note**
  >
  > 如果您选择不定义最小值或最大值，则可以省略 m 或 n。

- |：布尔值“Or”，与竖线两侧的字词相匹配。例如：
  - %gray|grey% 可以匹配“gray”或“grey”
  - %^starting|^initializing|^shutting down% 可以匹配“starting ...”或“initializing ...”或“shutting down”，但与“kipping initializing ...”不匹配

  - %abcc|ab[^c]$%可以匹配 “abcc...” 和 “aba”，但无法匹配 “aac...”
- \：转义字符，允许您使用运算符的字面含义而不是其特殊含义。例如，%\[.\]% 匹配任何由“[”和“]”括起来的单个字符，因为方括号已转义，例如“[a]”、“[b]”、“[7]”、“[@]”、“[]]”和“[ ]”。

  > **Note**
  >
  > %10\.10\.0\.1% 是创建正则表达式来匹配 IP 地址 10.10.0.1 的正确方法。

- *：匹配前面字词的零个或多个实例。例如， %ab*c% 可以匹配“ac”、“abc”和“abbbc”；%ab[0-9]*% 可以匹配“ab”、“ab0”和“ab129”。
- +：匹配前面字词的一个或多个实例。例如，%ab+c% 可以匹配“abc”、“abbc”和“abbbc”，但不匹配“ac”。
- .：匹配任意单个字符。例如，%.at% 匹配以“at”结尾的任意三个字符串，包括“hat”、“cat”、“bat”、“4at”、“#at”和“ at”（以空格开头）。

  > **Note**
  >
  > 在创建正则表达式以匹配 IP 地址时，转义 . 运算符很重要。例如，%10.10.0.1% 可能匹配“10010,051”，这可能不是表达式的实际预期用途。

- \d,\D: 匹配一个 digit/non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 character/non-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: 匹配字母数字和字母数字 character/non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，它表示以下字符表示 ASCII 的十六进制值。 hh指定指向 ASCII 表中字符的两个十六进制数字（0-9 和 A-F）。

> **Note**
>
> 您可以使用 \xhh 匹配筛选条件模式不支持的符号字符。例如，%\x3A% 匹配：；%\x28% 匹配（。

使用筛选条件模式将字词与正则表达式（regex）进行匹配

使用正则表达式匹配字词

您可以使用 % 括起来的正则表达式模式（正则表达式模式前后有百分号）来匹配日志事件中的字词。下面的代码段显示了筛选条件模式的示例，该模式返回包含 AUTHORIZED 关键字的所有日志事件。

有关支持的正则表达式的列表，请参阅支持的正则表达式。

```
%AUTHORIZED%
```

此筛选条件模式返回日志事件消息，例如以下内容：

- [ERROR 401] UNAUTHORIZED REQUEST
- [SUCCESS 200] AUTHORIZED REQUEST

使用筛选条件模式匹配非结构化日志事件中的字词

匹配非结构化日志事件中的字词

以下示例包含代码段，这些代码段显示了如何使用筛选条件模式匹配非结构化日志事件中的字词。

> **Note**
>
> 筛选条件模式区分大小写。将包含非字母数字字符的精确短语和字词用双引号括起来（""）。

Example: Match a single term

下面的代码段显示了单个字词筛选条件模式的示例，该模式返回其中的消息包含字词 ERROR 的所有录入事件。

```
ERROR
```

此筛选条件模式匹配日志事件消息，例如以下内容：

- `[ERROR 400] BAD REQUEST`
- `[ERROR 401] UNAUTHORIZED REQUEST`
- `[ERROR 419] MISSING ARGUMENTS`
- `[ERROR 420] INVALID ARGUMENTS`

Example: Match multiple terms

下面的代码段显示了多个字词筛选条件模式的示例，该模式返回其中的消息包含字词 ERROR 和 ARGUMENTS 的所有录入事件。

```
ERROR ARGUMENTS
```

筛选条件返回录入事件消息，例如以下内容：

- `[ERROR 419] MISSING ARGUMENTS`
- `[ERROR 420] INVALID ARGUMENTS`

筛选条件模式不会返回以下日志事件消息，因为它们不包含筛选条件模式中指定的两个字词。

- `[ERROR 400] BAD REQUEST`

- [ERROR 401] UNAUTHORIZED REQUEST

Example: Match optional terms

您可以使用模式匹配来创建返回包含可选字词的日志事件的筛选条件模式。在您想匹配的字词前放置问号（"?"）。下面的代码片段将显示筛选条件模式的一个示例，它将返回消息中包含单词 ERROR 或 ARGUMENTS 的所有日志事件。

```
?ERROR ?ARGUMENTS
```

此筛选条件模式匹配日志事件消息，例如以下内容：

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST
- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

> **Note**
>
> 不能将问号（“?”）与其他筛选条件模式组合，例如包含和排除术语。如果将“?”与其他筛选模式结合使用，则所有带问号的字词都将被忽略。
> 例如，以下筛选模式匹配所有包含词语 REQUEST 的事件，但带有问号（“?”）的筛选词将被忽略且无效。
>
> ```
> ?ERROR ?ARGUMENTS REQUEST
> ```
>
> 日志事件匹配项
>
> - [INFO] REQUEST FAILED
> - [WARN] UNAUTHORIZED REQUEST
> - [ERROR] 400 BAD REQUEST

Example: Match exact phrases

下面的代码段显示了筛选条件模式的示例，该模式返回其中的消息包含精确短语 INTERNAL SERVER ERROR 的录入事件。

```
"INTERNAL SERVER ERROR"
```

此筛选条件模式返回以下日志事件消息：

- `[ERROR 500] INTERNAL SERVER ERROR`

Example: Include and exclude terms

您可以创建筛选条件模式，该模式返回录入事件，其中的消息包含一些字词并排除其他字词。在您想排除的字词前放置减号（"-"）。下面的代码段显示了筛选条件模式的示例，该模式返回其中的消息包含字词 ERROR 并排除字词 ARGUMENTS 的录入事件。

```
ERROR -ARGUMENTS
```

此筛选条件模式返回日志事件消息，例如以下内容：

- `[ERROR 400] BAD REQUEST`
- `[ERROR 401] UNAUTHORIZED REQUEST`

此筛选条件模式不会返回以下日志事件消息，因为它们包含单词 ARGUMENTS。

- `[ERROR 419] MISSING ARGUMENTS`
- `[ERROR 420] INVALID ARGUMENTS`

Example: Match everything

您可以用双引号匹配录入事件中的所有内容。下面的代码段显示了筛选条件模式的示例，该模式返回所有录入事件。

```
" "
```

使用筛选条件模式匹配 JSON 日志事件中的字词

为 JSON 日志事件编写筛选条件模式

下面介绍了如何编写与包含字符串和数值的 JSON 字词匹配的筛选条件模式的语法。

Writing filter patterns that match strings

您可以创建筛选条件模式以匹配 JSON 日志事件中的字符串。以下代码段显示了基于字符串的筛选条件模式的语法示例。

```
{ PropertySelector EqualityOperator String }
```

用大括号 (“{}”) 将滤镜图案括起来。 String-based 过滤器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

您可以使用方括号表示法创建筛选模式来匹配包含句点 (.) 的 JSON 属性名称。将属性名称用单引号 (') 括起来。括号表示法中不支持双引号 (")，这会导致过滤器模式因语法错误而失败。以下筛选模式与该属性相匹配`cluster.name`。

```
{ $.['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。

Space-delimited 记录事件

```
INFO 09/25/2014 12:00:00 GET /service/resource/67 1200
INFO 09/25/2014 12:00:01 POST /service/resource/67/part/111 1310
WARNING 09/25/2014 12:00:02 Invalid user request
ERROR 09/25/2014 12:00:02 Failed to process request
```

Example: Match terms in order

以下以空格分隔的筛选条件模式会返回日志事件，其中事件中的第一个单词是 ERROR。

```
[w1=ERROR, w2]
```

> **Note**
>
> 创建使用模式匹配的以空格分隔的筛选条件模式时，必须在指定字词顺序后包含空白指示符。例如，如果您创建了一个筛选模式，其返回第一个单词为 ERROR 的日志事件，请在 w1 字词后包括空白 w2 指示符。

Example: Match terms with AND (&&) and OR (||)

您可以使用逻辑运算符 AND（“&&”）和 OR（“||”）创建包含条件的以空格分隔的筛选条件模式。以下筛选条件模式会返回日志事件，其中事件中的第一个单词是 ERROR 或 WARNING。

```
[w1=ERROR || w1=WARNING, w2]
```

Example: Exclude terms from matches

您可以创建以空格分隔的筛选条件模式，该筛选条件模式会返回排除一个或多个字词的日志事件。在您想排除的字词前放置不等号（"!="）。以下代码段显示了筛选条件模式的示例，该筛选条件模式返回第一个字词不是 ERROR 和 WARNING 的日志事件。

```
[w1!=ERROR && w1!=WARNING, w2]
```

Example: Match the top level item in a resource URI

以下代码段显示了筛选条件模式的示例，该筛选条件模式使用正则表达式匹配资源 URI 中的顶级项目。

```
[logLevel, date, time, method, url=%/service/resource/[0-9]+$%, response_time]
```

Example: Match the child level item in a resource URI

以下代码段显示了筛选条件模式的示例，该筛选条件模式使用正则表达式匹配资源 URI 中的子级项目。

```
[logLevel, date, time, method, url=%/service/resource/[0-9]+/part/[0-9]+$%,
 response_time]
```

启用从的日志记录 AWS 务

使用可搜索的目录查找 AWS 服务并打开其日志设置指南。有关每项服务的目标和权限模型的更多信息，请参阅支持的日志目标和权限。

许多服务仅向日志发布日 CloudWatch 志，但其他服务则使用预付费日志传输将日志直接发送到亚马逊简单存储服务或 Amazon Data Firehose。当您的主要要求是在其中一个目的地进行长期存储或处理时，直接配送非常有用。

即使您将日志直接发布到 Amazon S3 或 Firehose，也需要支付 CloudWatch 运费。如果您向 Amazon S3 发送日志，则*AWS_REGION*-S3-Egress-Bytes费用将显示在 Cost Explorer 或账单上。如果您向 Firehose 发送日志，则会显示*AWS_REGION*-FH-Egress-Bytes费用。有关销售日志定价的更多信息，请参阅 Amazon CloudWatch Pricin g 中的“日志”选项卡。

有些服务需要额外的权限才能传送日志。如果没有这些权限，日志传输就会失败。在比较表中，使用原始权限模型的服务被标记为支持（V1 权限）。使用当前模型的服务被标记为支持（V2 权限）。每个标签都链接到所需的策略。

<integration-catalog>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>

<integration></integration>
<integration></integration>
<integration></integration>

<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
<integration></integration>
</integration-catalog>

支持的日志目标和权限

下表显示了每项 AWS 服务支持哪些目标发送日志，以及需要哪个权限版本。

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
亚马逊 API Gateway 访问日志	Vended logs（已出售日志）	支持（V1 权限）			
应用程序负载均衡器访问日志	Vended logs（已出售日志）	支持的 [V2 权限]	支持的 [V2 权限]	支持的 [V2 权限]	
AWS AppSync logs	Custom logs（自定义日志）	支持			
亚马逊 Aurora MySQL 日志	Custom logs（自定义日志）	支持			
Amazon Bedrock 知识库日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
Amazon Bedrock 代理日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
Amazon Bedrock AgentCore 运行时	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）
Amazon Bedrock AgentCore 网关	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）
Amazon Bedrock AgentCore 身份	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）
Amazon Bedrock AgentCore 内存	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）
Amazon Bedrock AgentCore 付款	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
Amazon Bedrock AgentCore 工具	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）
Amazon Chime 媒体质量指标日志和 SIP 消息日志	Vended logs（已出售日志）	支持（V1 权限）			
CloudFront: 访问日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
AWS CloudHSM 审核日志	Custom logs（自定义日志）	支持			
CloudWatch 显然是评估事件日志	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）		
CloudWatch 互联网监视器日志	Vended logs（已出售日志）		支持（V1 权限）		

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
the section called “为 Lo CloudWatch gs Insights 查询执行日志创建传送源”	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
CloudTrail logs	Custom logs（自定义日志）	支持			
AWS CodeBuild logs	Custom logs（自定义日志）	支持			
Amazon CodeWhisperer 事件日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
Amazon Cognito logs	Vended logs（已出售日志）	支持（V1 权限）			
Amazon Connect 客户日志	Custom logs（自定义日志）	支持			

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
AWS DataSync logs	Custom logs（自定义日志）	支持			
AWS DevOps 代理 logs	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
亚马逊 ElastiCache （Redis OSS）日志	Vended logs（已出售日志）	支持（V1 权限）		支持（V1 权限）	
AWS Elastic Beanstalk logs	Custom logs（自定义日志）	支持			
Amazon 弹性容器服务日志	Custom logs（自定义日志）	支持			
亚马逊 Elastic Kubernetes Service 自动模式日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
亚马逊 Elastic Kubernetes Service 能力日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
亚马逊 Elastic Kubernetes Service 控制平面日志	Vended logs（已出售日志）	支持			
AWS Elemental MediaPackage 访问日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
AWS Elemental MediaTailor logs	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
AWS Entity Resolution 数据匹配服务 logs	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
Amazon EventBridge 管道记录	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持（V1 权限）	

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
Amazon EventBridge 活动巴士	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
Amazon EventBridge 事件总线（自定义）日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
AWS Fargate logs	Custom logs（自定义日志）	支持			
AWS Fault Injection Service 实验日志	Vended logs（已出售日志）		支持（V1 权限）		
Amazon FinSpace	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持（V1 权限）	
AWS Global Accelerator 流日志	Vended logs（已出售日志）		支持（V1 权限）		

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
AWS Glue 作业日志	Custom logs（自定义日志）	支持			
IAM 身份中心错误日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
Amazon 互动视频服务聊天记录	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持（V1 权限）	
AWS IoT logs	Custom logs（自定义日志）	支持			
AWS IoT FleetWise logs	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持（V1 权限）	
AWS Lambda logs	Vended logs（已出售日志）	支持	支持	支持	

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
亚马逊 Macie 日志	Custom logs（自定义日志）	支持			
AWS Mainframe Modernization	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持（V1 权限）	
适用于 Prometheus 的亚马逊托管服务日志	Vended logs（已出售日志）	支持（V1 权限）			
亚马逊 MSK 代理日志	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持（V1 权限）	
亚马逊 MSK Connect 日志	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持（V1 权限）	
亚马逊 MQ 日志	Custom logs（自定义日志）	支持			

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
AWS Network 防火墙日志	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持（V1 权限）	
AWS Network Fireall 代理日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
Network Load Balancer 访问日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
OpenSearch logs	Custom logs（自定义日志）	支持			
Amazon OpenSearch 服务摄取日志	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持（V1 权限）	
AWS PCS logs	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
亚马逊 Q Business 连接器日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
Amazon Q Business 对话日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
Amazon Q in Connect 人工智能代理事件日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
Amazon 快速日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
亚马逊关系数据库服务 PostgreSQL 日志	Custom logs（自定义日志）	支持			
亚马逊 Route 53 全球解析器查询日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
亚马逊 Route 53 公有 DNS 查询日志	Vended logs（已出售日志）	支持			
亚马逊 Route 53 解析器查询日志	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）		
AWS RTB Fabric 日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
AWS Security Hub CSPM	Vended logs（已出售日志）	支持（V2 权限）			
AWS Security Hub	Vended logs（已出售日志）	支持（V2 权限）			
AWS Shield 高级攻击流日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
Amazon 简单存储服务服务器访问日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
亚马逊 SageMaker AI 活动	Vended logs（已出售日志）	支持（V1 权限）			
亚马逊 SageMaker AI 工作者活动	Vended logs（已出售日志）	支持（V1 权限）			
AWS Site-to-Site VPN 日志	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持（V1 权限）	
Amazon 简单电子邮件服务日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
Amazon 简单通知服务日志	Custom logs（自定义日志）	支持			

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
Amazon 简单通知服务数据保护策略日志	Custom logs（自定义日志）	支持			
EC2 竞价型实例数据源文件	Vended logs（已出售日志）		支持（V1 权限）		
AWS Step Functions 快速工作流程和标准工作流程日志	Vended logs（已出售日志）	支持（V1 权限）			
Storage Gateway 审核日志和运行状况日志	Vended logs（已出售日志）	支持（V1 权限）			
AWS Transfer Family logs	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持（V1 权限）	
AWS Verified Access logs	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持（V1 权限）	

日志源	日志类型	the section called “发送到日志的 CloudWatch 日志”	the section called “发送到 Amazon S3 的日志”	the section called “日志发送至 Firehose”	the section called “追踪已发送至 X-Ray”
亚马逊 Virtual Private Cloud 流日志	Vended logs（已出售日志）	支持	支持（V1 权限）	支持（V1 权限）	
亚马逊 VPC 莱迪思访问日志	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持（V1 权限）	
Amazon VPC Route Server	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（V2 权限）	
AWS WAF logs	Vended logs（已出售日志）	支持（V1 权限）	支持（V1 权限）	支持	
Amazon WorkMail 审核日志	Vended logs（已出售日志）	支持（V2 权限）	支持（V2 权限）	支持（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。要使上表中列出的 AWS 服务能够将其日志发送到这些目的地，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。对于跨账户传输，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。

以下主题提供了每个目标的更多详细信息。

主题

- 发送到日志的 CloudWatch 日志
- 发送到 Amazon S3 的日志
- 日志发送至 Firehose

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**
>
> 当您指定 `logs:DescribeLogGroups`、`logs:DescribeResourcePolicies` 或 `logs:PutResourcePolicy` 权限时，请确保将其 `Resource` 行的 ARN 设置为使用 *

通配符，而不是仅指定单个日志组名称。例如，"Resource": "arn:aws:logs:us-east-1:111122223333:log-group:*"

如果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: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。对于新创建的订阅，资源策略是在日志组级别配置的，其最大大小为 51,200 字节。如果现有的账户级资源策略已经通过通配符授予权限，则不会创建单独的日志组级别策略。要检查特定日志组的日志Group-level 资源策略，请使用将--resource-arn参数设置为日志组 ARN，--policy-scope参数设置为的describe-resource-policies命令。RESOURCE

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite20150319",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "delivery.logs.amazonaws.com"
                ]
            },
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group:log-stream:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
```

```
                    "0123456789"
                ]
            },
            "ArnLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:us-east-1:111122223333:*"
                ]
            }
        }
    }
  ]
}
```

日志组的资源策略限制为 51,200 字节。一旦达到此限制，AWS 就无法添加新权限。
这要求客户手动修改策略，以授予delivery.logs.amazonaws.com服务主体
对logs:CreateLogStream和logs:PutLogEvents操作的权限。客户应使用带有通配符的日志组名称前缀，例如/aws/vendedlogs/*并在将来创建 Future Delivery 时使用此日志组名称。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite20150319",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "delivery.logs.amazonaws.com"
                ]
            },
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group/aws/
vendedlogs/*"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "aws:SourceAccount": [
                        "0123456789"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:111122223333:*"
                    ]
                }
            }
        }
    ]
}
```

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。

> **Important**
>
> 如果您要将日志发送到 Amazon S3 存储桶，并且存储桶策略包含 `NotAction` 或 `NotPrincipal` 元素，则自动向存储桶添加日志传输权限和创建日志订阅将会失败。为了成功创建日志订阅，您需要手动将日志传输权限添加到存储桶策略，然后创建日志订阅。有关更多信息，请参阅本节中的说明。

如果存储桶使用客户托管密 AWS KMS 钥进行服务器端加密，则您还必须为客户托管密钥添加密钥策略。有关更多信息，请参阅 Amazon S3。
如果目标存储桶已 SSE-KMS 启用存储桶密钥，则附加的客户托管 KMS 密钥策略将不再按预期适用于所有请求。有关更多信息，请参阅 SSE-KMS 使用 Amazon S3 存储桶密钥降低成本。
如果您使用的是带有客户托管 AWS KMS 密钥的销售日志和 S3 加密，则在配置存储桶时必须使用完全限定的密 AWS KMS 钥 ARN 而不是密钥 ID。有关更多信息，请参阅 put-bucket-encryption。

用户权限

若要能够设置首次将这些类型日志中的任一种日志发送到 Amazon S3，您必须登录具有以下权限的账户。

- `logs:CreateLogDelivery`
- `S3:GetBucketPolicy`
- `S3:PutBucketPolicy`

如果其中任何一种类型的日志已被发送到 Amazon S3 存储桶，要设置将其中另一种日志也发送到同一存储桶，您只需要 `logs:CreateLogDelivery` 权限。

S3 存储桶资源策略

接收日志的 S3 存储桶必须具有包含特定权限的资源策略。如果存储桶当前没有资源策略，而且设置日志记录的用户对存储桶具有 `S3:GetBucketPolicy` 和 `S3:PutBucketPolicy` 权限，那么当您开始将日志发送到 Amazon S3 时， AWS 会自动为其创建以下策略。

JSON

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

```
            },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "0123456789"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:111122223333:*"
                    ]
                }
            }
        },
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/AWSLogs/account-ID/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": [
                        "0123456789"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:111122223333:*"
                    ]
                }
            }
        }
    ]
}
```

在之前的策略中，对于 aws:SourceAccount，请指定将日志传送到此存储桶的账户 ID 列表。对于 aws:SourceArn，请按 arn:aws:logs:*source-region*:*source-account-id*:* 格式指定生成日志的资源 ARN 列表。

如果存储桶具有资源策略，但该策略不包含上一个策略中所示的语句，并且设置日志记录的用户对存储桶具有 S3:GetBucketPolicy 和 S3:PutBucketPolicy 权限，则该语句将附加到存储桶的资源策略中。

> **Note**
>
> 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

您可以使用 Amazon 密钥 () 启用服务器端加密，或者使用存储在 (SSE-S3) 中的 S3-managed 密钥启用服务器端加密，从而保护 Amazon S3 存储桶中的 AWS Key Management Service 数据。 AWS KMS SSE-KMS有关更多信息，请参阅使用服务器端加密保护数据。

如果您选择 SSE-S3，则无需进行其他配置。Amazon S3 处理加密密钥。

> **Warning**
>
> 如果选择 SSE-KMS，则必须使用客户托管密钥，因为这种情况下不支持使用 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。

如果选择 SSE-KMS，则必须使用客户托管密钥，因为这种情况下不支持使用 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。

日志发送至 Firehose

当前一部分的表中列出的日志类型发送到 Firehose 时，本部分适用：

用户权限

若要能够设置首次将这些类型日志中的任一种日志发送到 Firehose，您必须登录具有以下权限的账户。

- logs:CreateLogDelivery
- firehose:TagDeliveryStream

- iam:CreateServiceLinkedRole

如果其中任何一种日志已被发送到 Firehose，要设置将其中另一种日志也发送到 Firehose，您只需要 logs:CreateLogDelivery 和 firehose:TagDeliveryStream 权限。

用于权限的 IAM 角色

由于 Firehose 不使用资源策略， 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。

JSON

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

此服务相关角色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：

JSON

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

需要额外权限 [V2] 的日志记录

一些 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 以及 X-Ray 用于跟踪传输。

工作日志传输由三个元素组成：

- `DeliverySource`，这是代表实际发送日志的资源的逻辑对象。
- `DeliveryDestination`，这是代表实际传输目标的逻辑对象。
- `Delivery`，这连接传输源与传输目标

要在支持的 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。
- 如果您跨账户传送日志，则只能使用 Amazon S3 和 Firehose 目标。您必须在目标账户PutDeliveryDestinationPolicy中使用才能将 IAM 策略分配给目标。此策略授权创建从账户 A 中的传输源到账户 B 中的传输目标的传输。对于跨账户传输，您必须手动创建权限策略。有关设置示例，请参阅Cross-account 配送示例。
- 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。

日志传送设置示例

以下示例在同一个 AWS 账户中创建配送来源和配送目的地。将源资源 ARN 和日志类型替换为生成日志的服务支持的值。这些示例不需要配送目的地政策。

为 Lo CloudWatch gs Insights 查询执行日志创建传送源

以下示例为 L CloudWatch ogs Insights 查询执行日志创建了传输源。有关 ARN 的完整要求`logType`和资源，请参阅。PutDeliverySource

> **Note**
>
> 中的尾随通配符`log-group:*`是必需的，因为查询执行日志不绑定到一个日志组。 AWS 不支持此日志类型的特定日志组 ARN。

```
aws logs put-delivery-source \
    --name insights-query-logs \
    --resource-arn arn:aws:logs:region:account-id:log-group:* \
    --log-type INSIGHTS_QUERY_LOGS
```

创建交付来源

创建交付源，然后使用以下目标示例之一。

```
aws logs put-delivery-source \
    --name my-delivery-source \
    --resource-arn source-resource-arn \
    --log-type log-type
```

创建发送到 Lo CloudWatch gs 的传输

为现有日志组创建传送目标。

```
aws logs put-delivery-destination \
    --name my-cwl-delivery-destination \
    --delivery-destination-configuration \
    "destinationResourceArn=arn:aws:logs:region:account-id:log-group:log-group-name"
```

创建配送。

```
aws logs create-delivery \
    --delivery-source-name my-delivery-source \
    --delivery-destination-arn arn:aws:logs:region:account-id:delivery-destination:my-
cwl-delivery-destination
```

创建发往亚马逊 S3 的货件

为现有存储桶创建配送目的地。

```
aws logs put-delivery-destination \
    --name my-s3-delivery-destination \
    --delivery-destination-configuration \
    "destinationResourceArn=arn:aws:s3:::bucket-name"
```

创建配送。

```
aws logs create-delivery \
    --delivery-source-name my-delivery-source \
    --delivery-destination-arn arn:aws:logs:region:account-id:delivery-destination:my-
s3-delivery-destination
```

要配置目标前缀、后缀路径或路 Hive-compatible 径，请参见亚马逊 S3 对象密钥格式。

向 Firehose 创建配送

为现有传输流创建传 DirectPut 送目的地。

```
aws logs put-delivery-destination \
    --name my-firehose-delivery-destination \
    --delivery-destination-configuration \
    "destinationResourceArn=arn:aws:firehose:region:account-id:deliverystream/delivery-
stream-name"
```

创建配送。

```
aws logs create-delivery \
    --delivery-source-name my-delivery-source \
    --delivery-destination-arn arn:aws:logs:region:account-id:delivery-destination:my-
firehose-delivery-destination
```

创建配送至 X-Ray

要进行跟踪传输，请使用支持传送到的源服务和日志类型 X-Ray。创建逻辑 X-Ray 传送目的地。

```
aws logs put-delivery-destination \
    --name my-xray-delivery-destination \
    --delivery-destination-type XRAY
```

创建配送。

```
aws logs create-delivery \
    --delivery-source-name my-delivery-source \
    --delivery-destination-arn arn:aws:logs:region:account-id:delivery-destination:my-
xray-delivery-destination
```

要验证交付，请使用以下命令。

```
aws logs describe-deliveries \
    --delivery-source-name-prefix my-delivery-source
```

以下各节详细介绍了在登录期间使用 V2 流程将日志传输到每种类型目标时需要具备的权限。这些权限可以授予您登录时使用的 IAM 角色。

> **Important**
>
> 删除日志生成资源后，您有责任删除日志传输资源。为此，请按照以下步骤操作。
>
> 1. `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`DeliveryDestination`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
  - 用户权限
  - 日志组和资源策略
- 发送到 Amazon S3 的日志

  - 用户权限
  - 亚马逊 S3 存储桶资源政策
  - Amazon S3 存储桶服务器端加密
  - 亚马逊 S3 对象密钥格式
- 日志发送至 Firehose
  - 用户权限
  - 用于资源权限的 IAM 角色
- 追踪已发送至 X-Ray
  - 用户权限
  - X-Ray 资源政策
  - 启用事务搜索

发送到日志的 CloudWatch 日志

有关 AWS CLI 示例，请参阅创建发送到 Lo CloudWatch gs 的传输。

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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:GetDelivery",
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
                "logs:GetDeliveryDestinationPolicy",
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
```

```
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery",
                "logs:UpdateDeliveryConfiguration"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:111122223333:delivery:*",
            "arn:aws:logs:us-east-1:444455556666:delivery-source:*",
            "arn:aws:logs:us-east-1:777788889999:delivery-destination:*"
            ]
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries",
                "logs:DescribeConfigurationTemplates"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyCWL",
            "Effect": "Allow",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:123456789012:*"
            ]
        }
    ]
}
```

日志组和资源策略

接收日志的日志组必须具有包含特定权限的资源策略。如果日志组当前没有资源策略，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。对于新创建的订阅，资源策略是在日志组级别配置的，其最大大小为 51,200 字节。如果现有的账户级资源策略已经通过通配符授予权限，则不会创建单独的日志组级别策略。要检查特定日志组的日志Group-level资源策略，请使用将--resource-arn参数设置为日志组 ARN，--policy-scope参数设置为的describe-resource-policies命令。RESOURCE

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite20150319",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "delivery.logs.amazonaws.com"
                ]
            },
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group:log-
stream:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "0123456789"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:111122223333:*"
                    ]
                }
            }
        }
    ]
```

```
}
```

日志组的资源策略限制为 51,200 字节。一旦达到此限制，AWS 就无法添加新权限。
这要求客户手动修改策略，以授予delivery.logs.amazonaws.com服务主体
对logs:CreateLogStream和logs:PutLogEvents操作的权限。客户应使用带有通配符的日志组名称前缀，例如/aws/vendedlogs/*并在将来创建 Future Delivery 时使用此日志组名称。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite20150319",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "delivery.logs.amazonaws.com"
                ]
            },
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:my-log-group/aws/
vendedlogs/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "0123456789"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:111122223333:*"
                    ]
                }
            }
        }
```

```
    ]
}
```

发送到 Amazon S3 的日志

有关 AWS CLI 示例，请参阅创建发往亚马逊 S3 的货件。

用户权限

要启用向 Amazon S3 发送日志，您必须使用以下权限登录。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:GetDelivery",
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
                "logs:GetDeliveryDestinationPolicy",
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery",
                "logs:UpdateDeliveryConfiguration"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:111122223333:delivery:*",
    "arn:aws:logs:us-east-1:111122223333:delivery-source:*",
    "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
            ]
        },
        {
```

```
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries",
                "logs:DescribeConfigurationTemplates"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyS3",
            "Effect": "Allow",
            "Action": [
                "s3:PutBucketPolicy",
                "s3:GetBucketPolicy"
            ],
            "Resource": "arn:aws:s3:::bucket-name"
        }
    ]
}
```

亚马逊 S3 存储桶资源政策

接收日志的 S3 存储桶必须具有包含特定权限的资源策略。如果存储桶当前没有资源策略，而且设置日志记录的用户对存储桶具有 `S3:GetBucketPolicy` 和 `S3:PutBucketPolicy` 权限，那么当您开始将日志发送到 Amazon S3 时， AWS 会自动为其创建以下策略。

JSON

```
{
    "Version":"2012-10-17",
    "Id": "AWSLogDeliveryWrite20150319",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:PutObject",
```

```
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/AWSLogs/account-ID/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": [
                        "0123456789"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:111122223333:delivery-source:*"
                    ]
                }
            }
        }
    ]
}
```

在之前的策略中，对于 aws:SourceAccount，请指定将日志传送到此存储桶的账户 ID 列表。对于 aws:SourceArn，请按 arn:aws:logs:*source-region*:*source-account-id*:* 格式指定生成日志的资源 ARN 列表。

如果存储桶具有资源策略，但该策略不包含上一个策略中所示的语句，并且设置日志记录的用户对存储桶具有 S3:GetBucketPolicy 和 S3:PutBucketPolicy 权限，则该语句将附加到存储桶的资源策略中。

> **Note**
>
> 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

您可以通过启用服务器端加密来保护 Amazon S3 存储桶中的数据。您可以使用亚马逊 S3-managed 密钥 (SSE-S3) 或存储在 AWS Key Management Service (SSE-KMS) 中的 AWS KMS 密钥。有关更多信息，请参阅使用服务器端加密保护数据。

如果您选择 SSE-S3，则无需进行其他配置。Amazon S3 处理加密密钥。

> **需要客户管理的密钥**
>
> 如果选择 SSE-KMS，则必须使用客户管理的密钥。您不能使用 AWS 托管密钥。如果您使用 AWS 托管密钥配置加密，则 CloudWatch Logs 会以不可读的格式传送日志。

对于 SSE-KMS，在启用存储桶加密时指定密钥的 Amazon 资源名称 (ARN)。将以下内容添加到密钥策略（而不是您的 S3 存储桶策略中），以便日志传输账户可以写入您的 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": ["012345678901"]
        },
        "ArnLike": {
            "aws:SourceArn": ["arn:aws:logs:us-east-1:012345678901:delivery-source:*"]
        }
        }
}
```

对于aws:SourceAccount，请指定将日志传输到此存储桶的账户 ID。对于aws:SourceArn，请按以下格式指定传送源 ARN：arn:aws:logs:*source-region*:*source-account-id*:delivery-source:*。

亚马逊 S3 对象密钥格式

对于使用 V2 权限的交付，Amazon S3 对象密钥由目标前缀、日志类型、交付的后缀路径以及是否启用 Hive-compatible 路径确定。确切的服务定义路径和支持的后缀变量因日志类型而异。

目的地前缀

在调用存储桶 ARN 时附加到存储桶 ARN 的可选路径。PutDeliveryDestination例如arn:aws:s3:::*bucket-name*/*MyLogPrefix*。交付的对象以此前缀开头，后跟日志类型的服务定义路径，例如AWSLogs/*source-account-id*/*service-name*/。对于 CloudFront 标准日志记录 (v2)，目标前缀会替换默认路径，而不是放在默认路径之前。

后缀路径

您在其中为单个交付配置的可选路径S3DeliveryConfiguration。后缀可以包含静态文本和变量。要查找日志类型支持的变量，请致电DescribeConfigurationTemplates并查看allowedSuffixPathFields。如果未指定后缀路径，则在日志类型的默认后缀路径可用时使用该日志类型的默认后缀路径。

Hive-compatible 路径

如果enableHiveCompatiblePath是true，则显式中支持的变量将suffixPath呈现为*key*=*value*。对于诸如 Application Load Balancer 访问日志之类的日志类型，此格式还会更改服务定义的账户段。例如，AWSLogs/*source-account-id*/ 改为 AWSLogs/aws-account-id=*source-account-id*/。

如果省略suffixPath，则仅当日志类型的默认后缀路径支持时， Hive-compatible 格式才适用。例如，Application Load Balancer 访问日志支持这种格式。其他日志类型（包括 Amazon S3 服务器访问日志）以非 Hive 形式保留其原始默认后缀，包括资源和日期目录。

对于默认不支持 Hive-compatible格式化的日志类型，请指定suffixPath包含支持变量的有效日志类型。显式后缀取代默认后缀。使用allowedSuffixPathFields来自的日志类型，包括所需的资源和日期变量DescribeConfigurationTemplates。

以下示例显示了中账户111122223333的 Application Load Balancer 访问日志对象密钥的开头。us-east-1除非另有说明，否则这些示例假设没有目标前缀。

配置	对象密钥的开头
Hive-compatible 路径已禁用，后缀省略	`AWSLogs/111122223333/elasticloadbalancing/us-east-1/2026/09/10/`
Hive-compatible 路径已启用，后缀省略	`AWSLogs/aws-account-id=111122223333/elasticloadbalancing/region=us-east-1/year=2026/month=09/day=10/`
Hive-compatible 路径已启用，后缀 `myFolder/{yyyy}/{MM}/{dd}`	`AWSLogs/aws-account-id=111122223333/elasticloadbalancing/myFolder/year=2026/month=09/day=10/`
Hive-compatible 路径已启用，后缀 `myFolder/`	`AWSLogs/111122223333/elasticloadbalancing/myFolder/`
目标前缀`MyLogPrefix` 、 Hive-compatible 路径禁用、后缀省略	`MyLogPrefix/AWSLogs/111122223333/elasticloadbalancing/us-east-1/2026/09/10/`

静态后缀路径和存储桶权限

对于诸如 Application Load Balancer 访问日志之类的日志类型`suffixPath`，仅包含静态文本的无法 Hive-compatible 格式化服务定义的账户前缀。例如，`myFolder/`使用表中显示的普通账户前缀。如果 CloudWatch Logs 尝试在普通账户前缀下写入，而存储桶策略仅允许使用该 Hive-compatible 前缀，则交付失败。

当您创建`enableHiveCompatiblePath`设置为的 Application Load Balancer 交付时`true`，自动创建的存储桶策略会`s3:PutObject`对 Hive-compatible 前缀进行授权。

在日志类型`allowedSuffixPathFields`中至少包含一个支持的变量`suffixPath`，或者`suffixPath`对于默认后缀路径支持 Hive-compatible 格式化的日志类型，则省略该变量。CloudFront 标准日志 (v2) 的默认前缀格式与中的`suffixPath`变量无关。

Note

CloudFront 在 “将日志发送到 Amazon S3” 中记录了其标准日志 (v2) 路径行为和示例。

> **Note**
>
> 更新后缀路径或 Hive-compatible 设置会影响配置传播后的后续交付。现有对象不会被移动。当有效配送引用该目的地时，您无法更改目的地的 ARN（包括其前缀）。
> 存储桶策略必须允许使用`s3:PutObject`生成的前缀。管理存储桶策略时，请保留 Amazon S3 存储桶策略中显示的`aws:SourceAccount`和`aws:SourceArn`条件亚马逊 S3 存储桶资源政策，并且仅向所需的前缀授予访问权限。

日志发送至 Firehose

有关 AWS CLI 示例，请参阅向 Firehose 创建配送。

用户权限

要启用向 Firehose 发送日志，您必须使用以下权限登录。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:GetDelivery",
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
                "logs:GetDeliveryDestinationPolicy",
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery",
                "logs:UpdateDeliveryConfiguration"
            ],
            "Resource": [
```

```
            "arn:aws:logs:us-east-1:111122223333:delivery:*",
    "arn:aws:logs:us-east-1:111122223333:delivery-source:*",
    "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
            ]
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries",
                "logs:DescribeConfigurationTemplates"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyFH",
            "Effect": "Allow",
            "Action": [
                "firehose:TagDeliveryStream"
            ],
            "Resource": [
            "arn:aws:firehose:us-east-1:111122223333:deliverystream/*"
            ]
        },
        {
            "Sid": "CreateServiceLinkedRole",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery"
        }
    ]
}
```

用于资源权限的 IAM 角色

由于 Firehose 不使用资源策略，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。

JSON

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

此服务相关角色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：

JSON

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

```
  ]
}
```

追踪已发送至 X-Ray

有关 AWS CLI 示例，请参阅创建配送至 X-Ray。

用户权限

要启用向发送跟踪 AWS X-Ray，您必须使用以下权限登录。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccessForLogDeliveryActions",
            "Effect": "Allow",
              "Action": [
                "logs:GetDelivery",
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
                "logs:GetDeliveryDestinationPolicy",
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery",
                "logs:UpdateDeliveryConfiguration"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:111122223333:delivery:*",
            "arn:aws:logs:us-east-1:111122223333:delivery-source:*",
            "arn:aws:logs:us-east-1:111122223333:delivery-destination:*"
            ]
        },
        {
```

```
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries",
                "logs:DescribeConfigurationTemplates"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyXRay",
            "Effect": "Allow",
            "Action": [
                "xray:PutResourcePolicy",
                "xray:ListResourcePolicies",
                "xray:GetTraceSegmentDestination"
            ],
            "Resource": "*"
        }
    ]
}
```

X-Ray 资源政策

接收跟踪的目标账户必须具有包含特定权限的资源策略。当设置跟踪的用户在账户中拥有`xray:PutResourcePolicy`和`xray:ListResourcePolicies`权限时，会在您开始向发送跟踪时 AWS 自动创建资源策略 X-Ray。创建的策略取决于源服务：

Amazon Bedrock AgentCore resources

AWS 为每种资源类型创建一个资源策略。该政策使用范围限于账户边界的通配符模式，涵盖账户中相同 Amazon Bedrock AgentCore 资源类型的所有资源。例如，如果启用了Amazon Bedrock AgentCore内存资源以进行跟踪传输，则该策略将涵盖该账户中的所有内存资源，包括将来创建的任何内存资源。

JSON

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

```
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "xray:PutTraceSegments",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ForAllValues:ArnLike": {
          "logs:LogGeneratingResourceArns": "arn:aws:bedrock-agentcore:us-
east-1:123456789012:memory/*"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:logs:us-east-1:123456789012:delivery-
source:*"
        }
      }
    }
  ]
}
```

其他 AWS 服务

对于支持跟踪交付的其他服务，请 AWS 创建范围限于特定源资源的资源策略。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "xray:PutTraceSegments",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
```

```
        },
        "ForAllValues:ArnLike": {
          "logs:LogGeneratingResourceArns": "arn:aws:bedrock:us-
east-1:123456789012:knowledge-base/KnowledgeBaseId"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:logs:us-east-1:123456789012:delivery-
source:xray-test"
        }
      }
    }
  ]
}
```

启用事务搜索

要启用向发送跟踪 X-Ray，必须启用交易搜索。

Service-specific 权限

除了前面部分列出的特定于目标的权限之外，一些服务还需要明确授权，以允许客户从其资源发送日志，以此作为额外安全层。此策略授权对在该服务中提供日志的资源AllowVendedLogDeliveryForResource执行操作。对于这些服务，请使用以下策略，将服务命名空间 (*service*)、区域 (*region*)、 AWS 账户 ID (*account-id*) 和资源类型 (*resource-type*) 替换为您的资源的相应值。并非每个提供日志的服务都使用此操作。在您授予访问权限的服务的 vended Logs 文档中确认所需的权限和值，包括AllowVendedLogDeliveryForResource操作是否适用。以下示例显示了 Amazon SES 自动售卖日志的政策。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ServiceLevelAccessForLogDelivery",
            "Effect": "Allow",
            "Action": [
                "ses:AllowVendedLogDeliveryForResource"
            ],
            "Resource": "arn:aws:ses:us-east-1:123456789012:resource-type/*"
```

```
        }
    ]
}
```

Console-specific 权限

除了前面部分列出的权限外，如果您使用控制台而不是 API 设置日志传输，则还需要以下额外权限：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowLogDeliveryActionsConsoleCWL",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:*"
            ]
        },
        {
            "Sid": "AllowLogDeliveryActionsConsoleS3",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Sid": "AllowLogDeliveryActionsConsoleFH",
            "Effect": "Allow",
            "Action": [
                "firehose:ListDeliveryStreams",
```

```
                "firehose:DescribeDeliveryStream"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Cross-account 配送示例

在此示例中，涉及两个账户。拥有日志生成资源的账户是账户 A，ID:*123456789012*，拥有日志消耗资源的账户是账户 B，ID:。*111122223333*

账户 A 想使用 ARN arn: aws: bedrock:: knowledge-base/ 在其账户中提供 Amazon Bedrock 知识库中的日志。*us-east-1 123456789012 kb-12345678*

对于此示例，账户 A 需要以下权限：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowVendedLogDeliveryForKnowledgeBase",
            "Effect": "Allow",
            "Action": [
                "bedrock:AllowVendedLogDeliveryForResource"
            ],
            "Resource": "arn:aws:bedrock:us-east-1:123456789012:knowledge-
base/XXXXXXXXXX"
        },
        {
            "Sid": "CreateLogDeliveryPermissions",
            "Effect": "Allow",
            "Action": [
                "logs:PutDeliverySource",
                "logs:CreateDelivery"
            ],
            "Resource": [
```

```
            "arn:aws:logs:us-east-1:123456789012:delivery-source:*",
        "arn:aws:logs:us-east-1:123456789012:delivery:*",
        "arn:aws:logs:us-east-1:444455556666:delivery-destination:*"
            ]
        }
    ]
}
```

创建传输源

首先，账户 A 使用其 bedrock 知识库创建传输源：

```
aws logs put-delivery-source --name my-delivery-source --log-type APPLICATION_LOGS --
resource-arn arn:aws:bedrock:region:AAAAAAAAAAAA:knowledge-base/XXXXXXXXXX
```

接下来，账户 B 必须使用以下流之一创建传输目标：

- 配置传输到 Amazon S3 存储桶
- 配置向 Firehose 流的传输

配置传输到 Amazon S3 存储桶

用户 B 希望使用 ARN arn:aws:s3:::amzn-s3-demo-bucket 将日志接收到其 S3 存储桶中。对于此示例，账户 B 将需要以下权限：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PutLogDestinationPermissions",
            "Effect": "Allow",
            "Action": [
                "logs:PutDeliveryDestination",
                "logs:PutDeliveryDestinationPolicy"
            ],
            "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:*"
```

```
        }
    ]
}
```

存储桶在其存储桶策略中需要具有以下权限：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogsDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/AWSLogs/123456789012/
*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": [
                    "123456789012"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                    "arn:aws:logs:us-east-1:123456789012:delivery-source:my-
delivery-source"
                    ]
                }
            }
        }
    ]
}
```

如果存储桶使用加密 SSE-KMS，请确保 AWS KMS 密钥策略具有相应的权限。例如，如果 KMS 密钥是 arn:aws:kms:*us-east-1*:*111122223333*:key/*1234abcd-12ab-34cd-56ef-1234567890ab*，请使用以下内容：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowLogsGenerateDataKey",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                    "123456789012"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                    "arn:aws:logs:us-east-1:123456789012:delivery-source:my-
delivery-source"
                    ]
                }
            }
        }
    ]
}
```

然后，账户 B 可以创建一个以 S3 存储桶为目标资源的传输目标：

```
aws logs put-delivery-destination --name my-s3-delivery-destination --delivery-
destination-configuration "destinationResourceArn=arn:aws:s3:::amzn-s3-demo-bucket"
```

接下来，账户 B 在其新创建的传输目标上创建传输目标策略，该策略将授予账户 A 创建日志传输的权限。将添加到新创建的传输目标的策略如下：

JSON

```
    {
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateDelivery",
            "Effect": "Allow",
            "Principal": {
                "AWS": "123456789012"
            },
            "Action": [
                "logs:CreateDelivery"
            ],
            "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:amzn-s3-demo-bucket"
        }
    ]
}
```

此策略将以 destination-policy-s3.json 形式保存在账户 B 的计算机中。要附加此资源，账户 B 将运行以下命令：

```
aws logs put-delivery-destination-policy --delivery-destination-name my-s3-delivery-
destination --delivery-destination-policy file://destination-policy-s3.json
```

最后，账户 A 创建传输，将账户 A 中的传输源链接到账户 B 中的传输目标。

```
aws logs create-delivery --delivery-source-name my-delivery-source --delivery-
destination-arn arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:my-s3-delivery-
destination
```

配置向 Firehose 流的传输

在此示例中，账户 B 希望将日志接收到其 Firehose 流中。Firehose 直播具有以下 ARN，并且已配置为使用传输流类型： DirectPut

arn:aws:firehose:*us-east-1*:*111122223333*:deliverystream/*log-delivery-stream*

对于此示例，账户 B 需要以下权限：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowFirehoseCreateSLR",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery"
        },
        {
            "Sid": "AllowFirehoseTagging",
            "Effect": "Allow",
            "Action": [
                "firehose:TagDeliveryStream"
            ],
            "Resource": "arn:aws:firehose:us-
east-1:111122223333:deliverystream/X"
        },
        {
            "Sid": "AllowFirehoseDeliveryDestination",
            "Effect": "Allow",
            "Action": [
                "logs:PutDeliveryDestination",
                "logs:PutDeliveryDestinationPolicy"
            ],
            "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:*"
        }
```

```
    ]
}
```

Firehose 流必须将标签 LogDeliveryEnabled 设置为 true。

然后，账户 B 将创建一个以 Firehose 流为目标资源的传输目标：

```
aws logs put-delivery-destination --name my-fh-delivery-destination --delivery-
destination-configuration
 "destinationResourceArn=arn:aws:firehose:region:BBBBBBBBBBBB:deliverystream/X"
```

接下来，账户 B 在其新创建的传输目标上创建传输目标策略，该策略将授予账户 A 创建日志传输的权限。要添加到新创建的传输目标的策略如下：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateDelivery",
            "Effect": "Allow",
            "Principal": {
            "AWS": "123456789012"
            },
            "Action": [
                "logs:CreateDelivery"
            ],
            "Resource": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:amzn-s3-demo-bucket"
        }
    ]
}
```

此策略将以 destination-policy-fh.json 形式保存在账户 B 的计算机中。要附加此资源，账户 B 运行以下命令：

```
aws logs put-delivery-destination-policy --delivery-destination-name my-fh-delivery-
destination --delivery-destination-policy file://destination-policy-fh.json
```

最后，账户 A 创建传输，将账户 A 中的传输源链接到账户 B 中的传输目标。

```
aws logs create-delivery --delivery-source-name my-delivery-source --delivery-
destination-arn arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:my-fh-delivery-
destination
```

Cross-service 混乱的副手预防

混淆代理问题是一个安全性问题，即不具有某操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在中 AWS，跨服务模仿可能会导致混乱的副手问题。 Cross-service 当一个服务（调用服务）调用另一个服务（被调用的服务）时，可能会发生模仿行为。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访问您账户中资源的权限。

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 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。用于aws:SourceOrgPaths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对 CloudWatch 日志的访问权限，以便将数据写入 Kinesis Data Streams 和 Firehose步骤 1：创建目标，步骤 2：创建目标并显示如何使用aws:SourceArn全局条件上下文密钥来帮助防止混乱的副手问题。

使用遥测启用规则自动启用日志

您可以使用遥测启用规则为您的 AWS 资源自动配置日志收集。规则可帮助您标准化组织或账户中的日志收集，并确保一致的监控覆盖范围。

启用规则允许您：

- 自动为与您的规则范围匹配的新资源和现有资源启用日志记录
- 在组织、组织单位 (OU) 或个人账户级别应用规则
- 使用标签筛选规则影响哪些资源

启用规则目前支持以下 AWS 遥测源：亚马逊 VPC 流日志、日志、Route 53 解析器查询日 AWS WAF 志、NLB 访问日志、Amazon EKS 控制平面日志、 CloudTrail 数据和管理事件、亚马逊基岩日志、亚马逊 EC2 详细指标、Security AWS Hub、Amazon Bedrock Gateway、 AgentCore Amazon Bedrock Gat AgentCore eway、Amazon Bedrock 内存和分发。 AgentCore CloudFront

有关启用规则（包括规则行为、管理规则、故障排除和服务特定注意事项）的完整详细信息，请参阅 Amazon 用户指南中的遥测启用规则。 CloudWatch

规则评估层次结构

启用规则按层次进行评估：首先是组织规则，然后是规则，然后是账户级别的 OU-level 规则。更高级别的规则提供了基本的遥测数据。 Lower-level 规则可以添加额外的遥测数据，但不能减少遥测。如果同一作用域内存在冲突规则，则在冲突得到解决之前，不应用任何规则。

创建启用规则

创建遥测启用规则

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择遥测配置。
3. 选择启用规则选项卡。
4. 选择添加规则。
5. 指定规则名称、范围（组织、组织单位或帐户）、数据源和遥测类型。
6. （可选）添加标签以筛选规则影响的资源。
7. 选择 Create rule（创建规则）。

启用来自第三方来源的日志记录

CloudWatch 日志支持通过直接 API 集成和 Amazon S3 存储桶集成从第三方来源提取日志。您还可以通过 Security Hub CSPM 接收其他第三方 AWS 安全调查结果。

直接第三方集成

CloudWatch Logs 提供与以下第三方来源的直接集成。这些集成使用直接 API 连接或 Amazon S3 存储桶集成将日志提取到日志中： CloudWatch

- CrowdStrike 猎鹰
- Microsoft Office 365
- Okta Auth0
- Microsoft Entra ID
- 帕洛阿尔托网络 NGFW
- Microsoft Windows 事件
- Wiz
- Zscaler Internet Access
- Okta SSO
- SentinelOne 端点安全
- GitHub 审核日志
- ServiceNow CMDB
- Cisco Umbrella

有关设置说明，请参阅《Amazon CloudWatch 用户指南》中的第三方集成设置指南。

通过 S AWS ecurity Hub CSPM 提供的其他来源

除了直接集成外，第三方安全合作伙伴还会将调查结果发送给 Sec AWS urity Hub CSPM，然后这些结果作为日志中的数据源提供。 CloudWatch 下表列出了 Security Hub CSPM 合作伙伴集成及其集成类型。

要启用 Security Hub CSPM 结果作为 CloudWatch 日志中的数据源，请在控制台中为 Security Hub 创建遥测启用 AWS 规则。 CloudWatch 启用规则配置为自动将 CloudWatch 来自 Security Hub CSPM

的发现结果提取到托管日志组中。有关 step-by-step说明，请参阅 A mazon CloudWatch 用户指南中的遥测启用规则。

Partner	集成
3 CORESec — NTA	通过 Security Hub CSPM 发送调查结果
警报逻辑- SIEMless 威胁管理	通过 Security Hub CSPM 发送调查结果
Aqua Security — 云原生安全平台	通过 Security Hub CSPM 发送调查结果
水族安全 — Kube-Bench	通过 Security Hub CSPM 发送调查结果
盔甲 — 任何地方的盔甲	通过 Security Hub CSPM 发送调查结果
AttackIQ	通过 Security Hub CSPM 发送调查结果
梭子鱼网络 — 云安全守护者	通过 Security Hub CSPM 发送调查结果
BigID — BigID Enterprise	通过 Security Hub CSPM 发送调查结果
蓝色六边形	通过 Security Hub CSPM 发送调查结果
检查点 — CloudGuard IaaS	通过 Security Hub CSPM 发送调查结果
检查点 — CloudGuard 姿势管理	通过 Security Hub CSPM 发送调查结果
Claroty — xDome	通过 Security Hub CSPM 发送调查结果

Partner	集成
云存储安全 — 适用于 Amazon S3 的防病毒软件	通过 Security Hub CSPM 发送调查结果
对比安全 — 对比度评估	通过 Security Hub CSPM 发送调查结果
CrowdStrike — CrowdStrike 猎鹰	通过 Security Hub CSPM 发送调查结果
CyberArk — 特权威胁分析	通过 Security Hub CSPM 发送调查结果
数据定理	通过 Security Hub CSPM 发送调查结果
德拉塔	通过 Security Hub CSPM 发送调查结果
Forcepoint — CAS	通过 Security Hub CSPM 发送调查结果
Forcepoint — 云安全网关	通过 Security Hub CSPM 发送调查结果
原力点 — DLP	通过 Security Hub CSPM 发送调查结果
Forcepoint — NGFW	通过 Security Hub CSPM 发送调查结果
赋格曲	通过 Security Hub CSPM 发送调查结果
Guardicore — Centra	通过 Security Hub CSPM 发送调查结果
HackerOne — 漏洞情报	通过 Security Hub CSPM 发送调查结果

Partner	集成
JFrog — X 射线	通过 Security Hub CSPM 发送调查结果
瞻博网络 — vSRX 下一代防火墙	通过 Security Hub CSPM 发送调查结果
k9 安全 — 访问分析器	通过 Security Hub CSPM 发送调查结果
蕾丝工艺	通过 Security Hub CSPM 发送调查结果
McAfee — MVISION CNAPP	通过 Security Hub CSPM 发送调查结果
NETSCOUT — 网络调查员	通过 Security Hub CSPM 发送调查结果
Orca — 云安全平台	通过 Security Hub CSPM 发送调查结果
帕洛阿尔托网络 — Prisma 云计算	通过 Security Hub CSPM 发送调查结果
帕洛阿尔托网络 — Prisma 云企业版	通过 Security Hub CSPM 发送调查结果
Plerion — 云安全平台	通过 Security Hub CSPM 发送调查结果
徘徊者	通过 Security Hub CSPM 发送调查结果
Qualys — 漏洞管理	通过 Security Hub CSPM 发送调查结果
Rapid7 — InsightVM	通过 Security Hub CSPM 发送调查结果

Partner	集成
SentinelOne	通过 Security Hub CSPM 发送调查结果
Snyk	通过 Security Hub CSPM 发送调查结果
Sonrai 安全 — Sonrai Dig	通过 Security Hub CSPM 发送调查结果
Sophos — 服务器保护	通过 Security Hub CSPM 发送调查结果
StackRox — Kubernetes 安全	通过 Security Hub CSPM 发送调查结果
Sumo Logic — 机器数据分析	通过 Security Hub CSPM 发送调查结果
赛门铁克-云工作负载保护	通过 Security Hub CSPM 发送调查结果
Tenable.io	通过 Security Hub CSPM 发送调查结果
Trend Micro – Cloud One	通过 Security Hub CSPM 发送调查结果
Vectra — Cognito Detect	通过 Security Hub CSPM 发送调查结果
Wiz	通过 Security Hub CSPM 发送调查结果
Caveonix — Caveonix Cloud	通过 Security Hub CSPM 发送和接收调查结果
Cloud Custodian	通过 Security Hub CSPM 发送和接收调查结果

Partner	集成
DisruptOps	通过 Security Hub CSPM 发送和接收调查结果
Kion	通过 Security Hub CSPM 发送和接收调查结果
Turbot	通过 Security Hub CSPM 发送和接收调查结果

> **Note**
>
> 此列表反映了在撰写本文时发送调查结果的 Security Hub 合作伙伴集成。由于 S AWS ecurity Hub 会定期添加新的合作伙伴集成，因此请参阅 Sec urity Hub 用户指南中的第三方产品与 S AWS ecurity Hub CSPM 的集成，了解可用合作伙伴的大部分 up-to-date列表。

将日志数据导出到 Amazon S3

本章为您提供了信息，以便您可以将日志数据从日志组导出到 Amazon S3 存储桶进行自定义处理和分析，或加载到其他系统。您可以将同一账户或其他账户的 S3 存储桶导出到 S3 存储桶。

您可执行以下操作：

- 将日志数据导出到由 AWS Key Management Service () SSE-KMS AWS KMS中加密的 S3 存储桶
- 将日志数据导出到已启用 S3 对象锁定且具有保留期的 S3 桶

我们建议您不要定期将日志导出到 Amazon S3 以持续归档日志。对于该使用案例，我们建议您使用订阅。有关订阅的更多信息，请参阅Real-time 使用订阅处理日志数据。

要开始导出进程，您必须创建一个用于存储导出的日志数据的 S3 存储桶。您可以将已导出的文件存储在您的 S3 桶中，并定义 Amazon S3 生命周期规则以自动存档或删除已导出的文件。

您可以将使用 AES-256 或使用加密的存储桶导出到 S3 存储桶。 SSE-KMS不支持导出到使用加密 DSSE-KMS 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。要整理导出的数据，请为每个导出任务指定一个前缀，该前缀将用作所有导出对象的 Amazon S3 密钥前缀。例如，`prod/app-logs/2026-01-03/` 或 `log-group-name/backup/`

> **Note**
>
> Time-based 不能保证对导出文件中的日志数据块进行排序。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 Real-time 使用订阅处理日志数据。

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 (CLI)
- 取消导出任务 (CLI)

概念

开始之前，请熟悉以下导出概念：

日志组名称

与导出任务关联的日志组的名称。该日志组中的日志数据将导出至指定的 S3 桶中。

从（时间戳）

必填的时间戳，表示自 1970 年 1 月 1 日 00:00:00 UTC 以来的毫秒数。日志组中在此时间或之后提取的所有日志事件都将被导出。

至（时间戳）

必填的时间戳，表示自 1970 年 1 月 1 日 00:00:00 UTC 以来的毫秒数。日志组中该时间之前获得的所有日志事件都会被导出。

目标存储桶

与导出任务关联的 S3 桶的名称。此存储桶用于导出指定日志组中的日志数据。

目标前缀

可选属性，用作所有导出对象的 Amazon S3 键前缀。这有助于在您的存储桶中创建类似于文件夹的组织结构。

使用控制台将日志数据导出到 Amazon S3

在以下示例中，您使用亚马逊 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`amzn-s3-demo-bucket`。

支持将日志数据导出到由 SSE-KMS 加密的 S3 存储桶。不支持导出到使用加密 DSSE-KMS 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

- Same-account 导出（控制台）
- Cross-account 导出（控制台）

Same-account 导出（控制台）

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

- 创建 Amazon S3 存储桶（控制台）
- 设置访问权限（控制台）
- 在 Amazon S3 存储桶上设置权限（控制台）
- （可选）将使用 SSE-KMS （控制台）加密的 Amazon S3 存储桶导出到目标
- 创建导出任务（控制台）

创建 Amazon S3 存储桶（控制台）

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**
>
> Amazon S3 存储桶必须与要导出的日志数据位于同一区域。 CloudWatch 日志不支持将数据导出到其他区域的 Amazon S3 存储桶。

创建 Amazon 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。

设置访问权限（控制台）

要创建导出任务，您需要使用 AmazonS3ReadOnlyAccess IAM 角色和以下权限登录：

- `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。

- 通过身份提供者在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中针对第三方身份提供者创建角色（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。

在 Amazon S3 存储桶上设置权限（控制台）

默认情况下，所有 Amazon S3 存储桶和对象都是私有的。只有创建存储桶 AWS 账户 的资源所有者才能访问该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。

当您设置策略时，建议包含一个随机生成的字符串作为存储桶的前缀，这样只会将目标日志流导出到该存储桶。

> **Important**
>
> 为了使导出到 Amazon S3 存储桶更加安全，我们现在要求您指定允许将日志数据导出到 S3 桶的源账户列表。
> 在以下示例中，`aws:SourceAccount` 密钥中账户 ID 的列表是用户可以从其中将日志数据导出到 Amazon S3 存储桶的账户。`aws:SourceArn` 密钥是正在进行的操作的资源。如本示例所示，您可以将其限制为特定的日志组，也可以使用通配符。
> 建议您还包括创建 S3 桶的账户的账户 ID，以允许在同一账户内导出。

设置 Amazon S3 存储桶上的权限

1. 在 Amazon S3 控制台中，选择您创建的存储桶。
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 `amzn-s3-demo-bucket` 更改为 S3 存储桶的名称。请务必指定正确的区域端点，例如 Principal（委托人）的 `us-west-1`。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
      {
          "Sid": "AllowCloudWatchLogsGetBucketAcl",
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
          "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
          "Condition": {
```

```
            "StringEquals": {
                "aws:SourceAccount": [
                    "123456789012",
                    "111122223333"
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:123456789012:log-group:*",
                        "arn:aws:logs:us-east-1:111122223333:log-group:*"
                    ]
            }
        }
    },
    {
        "Sid": "AllowCloudWatchLogsPutObject",
        "Action": "s3:PutObject",
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
        "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control",
                "aws:SourceAccount": [
                    "123456789012",
                    "111122223333"
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:123456789012:log-group:*",
                        "arn:aws:logs:us-east-1:111122223333:log-group:*"
                    ]
            }
        }
    }
]
}
```

4. 选择 Save，将您刚添加的策略设置为存储桶上的访问策略。此策略允许 CloudWatch 日志将日志数据导出到您的 Amazon 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）将使用 SSE-KMS （控制台）加密的 Amazon S3 存储桶导出到目标

仅当您要导出到使用服务器端加密的 Amazon S3 存储桶时，才需要执行此步骤。 AWS KMS keys这种加密被称为 SSE-KMS。

导出到使用加密的存储桶 SSE-KMS

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. 对于密钥类型，选择对称。
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。完成后，请*Region*替换为日志所在区域，替换*account-ARN*为拥有 KMS 密钥的账户的 ARN。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow CWL Service Principal usage",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.Region.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "account-ARN"
            },
            "Action": [
                "kms:GetKeyPolicy*",
                "kms:PutKeyPolicy*",
                "kms:DescribeKey*",
                "kms:CreateAlias*",
                "kms:ScheduleKeyDeletion*",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

14. 选择保存更改。
15. 打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
16. 找到您在 创建 S3 存储桶 (CLI) 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 加密” 下，选择 “启用”。
19. 在加密类型下，选择AWS 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。

创建导出任务（控制台）

在此过程中，您将创建用于从日志组中导出日志的导出任务。

使用 CloudWatch 控制台将数据导出到 Amazon S3

1. 如 设置访问权限（控制台） 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 存储桶名称），选择一个 Amazon S3 存储桶。
10. 对于 S3 Bucket prefix（S3 存储桶前缀），输入在存储桶策略中指定的随机生成的字符串。
11. 选择 Export（导出），将日志数据导出到 Amazon S3。
12. 要查看您导出到 Amazon S3 的日志数据的状态，请依次选择 Actions（操作）、View all exports to Amazon S3（查看导出到 Amazon S3 的所有内容）。

Cross-account 导出（控制台）

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

- 创建用于跨账户导出的 Amazon S3 存储桶（控制台）

- 设置跨账户导出的访问权限（控制台）
- 在 S3 存储桶上设置跨账户导出权限（控制台）
- （可选）导出到目标 Amazon S3 存储桶加密后 SSE-KMS 用于跨账户导出（控制台）
- 为跨账户导出创建导出任务（控制台）

创建用于跨账户导出的 Amazon S3 存储桶（控制台）

我们建议您使用专为 Logs 创建的存储 CloudWatch 桶。但是，如果您想使用现有的存储桶，则可以跳过此过程。

> **Note**
>
> Amazon S3 存储桶必须与要导出的日志数据位于同一区域。 CloudWatch 日志不支持将数据导出到其他区域的 Amazon S3 存储桶。

创建 Amazon 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。

设置跨账户导出的访问权限（控制台）

首先，您必须创建新的 IAM 策略，以使 CloudWatch Logs 能够在目标账户中对目标 Amazon S3 存储桶`s3:PutObject`执行操作。

除了`s3:PutObject`操作外，策略中包含的其他操作还取决于目标存储桶是使用 AWS KMS 加密还是使用 S3 对象所有权设置启用 ACL。

- 如果使用 KMS 加密，请为密钥资源添加 `kms:GenerateDataKey` 和 `kms:Decrypt` 操作
- 如果在存储桶上启用了 ACL，请为存储桶资源添加 `s3:PutObjectAcl` 操作

在以下策略中更改amzn-s3-demo-bucket目标 S3 存储桶的名称。

创建 IAM policy 以将日志导出到 Amazon S3 存储桶

1. 使用 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。

   JSON

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

   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。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
```

```
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ]
}
```

如果在目标存储桶上启用了 ACL，则在上述策略中将 s3: PutObjectAcl 添加到s3: PutObject 操作块。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "s3:PutObject",
               "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

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。

要创建导出任务，您必须使用附加了AmazonS3ReadOnlyAccess托管策略、上面创建的 IAM 策略的 IAM 角色以及以下权限登录：

- 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。
- 通过身份提供者在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中针对第三方身份提供者创建角色（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。

在 S3 存储桶上设置跨账户导出权限（控制台）

默认情况下，所有 S3 桶和对象都是私有的。只有创建存储桶 AWS 账户 的资源所有者才能访问该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。

当您设置策略时，建议包含一个随机生成的字符串作为存储桶的前缀，这样只会将目标日志流导出到该存储桶。

> **Important**
>
> 为了使导出到 S3 桶更加安全，我们现在要求您指定允许将日志数据导出到 S3 桶的源账户列表。
> 在以下示例中，`aws:SourceAccount` 密钥中账户 ID 的列表会是用户可以从其中将日志数据导出到 S3 存储桶的账户。`aws:SourceArn` 密钥是正在进行的操作的资源。如本示例所示，您可以将其限制为特定的日志组，也可以使用通配符。
> 建议您还包括创建 S3 桶的账户的账户 ID，以允许在同一账户内导出。

设置 Amazon S3 存储桶上的权限

1. 在 Amazon S3 控制台中，选择您创建的存储桶。

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 amzn-s3-demo-bucket 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-east-1。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
          "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "aws:SourceAccount": [
                    "123456789012",
                    "111122223333"
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:123456789012:log-group:*",
                        "arn:aws:logs:us-east-1:111122223333:log-group:*"
                    ]
            }
          }
      },
      {
          "Action": "s3:PutObject",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
          "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control",
                "aws:SourceAccount": [
                    "123456789012",
                    "111122223333"
                ]
```

```
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:123456789012:log-group:*",
                        "arn:aws:logs:us-east-1:111122223333:log-group:*"
                    ]
            }
          }
      },
      {
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/role_name"
          },
          "Action": "s3:PutObject",
          "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
          "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control"
            }
          }
       }
    ]
}
```

4. 选择 Save，将您刚添加的策略设置为存储桶上的访问策略。此策略允许 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）导出到目标 Amazon S3 存储桶加密后 SSE-KMS 用于跨账户导出（控制台）

仅当您要导出到使用服务器端加密的 S3 存储桶时，才需要执行此过程。 AWS KMS keys这种加密被称为 SSE-KMS。

导出到使用加密的存储桶 SSE-KMS

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. 对于密钥类型，选择对称。
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。完成后，请*us-east-1*使用您的日志区域、拥*account-ARN*有 KMS 密钥的账户的 ARN、拥有 KMS 密钥的账号、*123456789012* kms-key ID 以及*role_name*用于创建导出任务的角色替换。*key_id*

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow CWL Service Principal usage",
            "Effect": "Allow",
            "Principal": {
            "Service": "logs.us-east-1.amazonaws.com"
            },
            "Action": [
```

```
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "account-ARN"
            },
            "Action": [
                "kms:GetKeyPolicy*",
                "kms:PutKeyPolicy*",
                "kms:DescribeKey*",
                "kms:CreateAlias*",
                "kms:ScheduleKeyDeletion*",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM Role Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/role_name"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-id"
        }
    ]
}
```

14. 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 创建 S3 存储桶 (CLI) 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 加密” 下，选择 “启用”。
19. 在加密类型下，选择AWS 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。

为跨账户导出创建导出任务（控制台）

在此过程中，您将创建用于从日志组中导出日志的导出任务。

使用 CloudWatch 控制台将数据导出到 Amazon S3

1. 如 设置访问权限（控制台） 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`amzn-s3-demo-bucket`。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。不支持导出到使用加密 DSSE-KMS 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

- Same-account 导出 (CLI)
- Cross-account 导出 (CLI)

Same-account 导出 (CLI)

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

- 创建 S3 存储桶 (CLI)
- 设置访问权限 (CLI)
- 在 S3 存储桶上设置权限 (CLI)
- （可选）将使用 SSE-KMS (CLI) 加密的 Amazon S3 存储桶导出到目标
- 创建导出任务 (CLI)

创建 S3 存储桶 (CLI)

我们建议您使用专为 Logs 创建的存储 CloudWatch 桶。但是，如果您想使用现有的存储桶，则可以跳过此过程。

> **Note**
>
> S3 桶必须与要导出的日志数据位于同一区域。 CloudWatch 日志不支持将数据导出到其他区域的 S3 存储桶。

要创建 S3 存储桶，请使用 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 amzn-s3-demo-bucket --create-bucket-configuration
 LocationConstraint=us-east-2
```

下面是示例输出。

```
{
    "Location": "/amzn-s3-demo-bucket"
}
```

设置访问权限 (CLI)

要稍后创建导出任务，您需要使用 AmazonS3ReadOnlyAccess IAM 角色和以下权限登录：

- 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。
- 通过身份提供者在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中针对第三方身份提供者创建角色（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。

在 S3 存储桶上设置权限 (CLI)

默认情况下，所有 S3 桶和对象都是私有的。仅资源所有者（创建了存储桶的账户）能够访问存储桶及其包含的任何对象。不过，资源所有者可以选择通过编写访问策略来向其他资源和用户授予访问权限。

> **Important**
>
> 为了使导出到 S3 桶更加安全，我们现在要求您指定允许将日志数据导出到 S3 桶的源账户列表。
> 在以下示例中，aws:SourceAccount 密钥中账户 ID 的列表会是用户可以从其中将日志数据导出到 S3 存储桶的账户。aws:SourceArn 密钥是正在进行的操作的资源。如本示例所示，您可以将其限制为特定的日志组，也可以使用通配符。
> 建议您还包括创建 S3 桶的账户的账户 ID，以允许在同一账户内导出。

在 S3 桶上设置权限

1. 创建一个名为 policy.json 的文件并添加以下访问策略，将 amzn-s3-demo-bucket 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-east-1。使用文本编辑器以创建此策略文件。请勿使用 IAM 控制台。

   JSON

```
{
"Version":"2012-10-17",
"Statement": [
  {
      "Sid": "AllowGetBucketAcl",
      "Action": "s3:GetBucketAcl",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
      "Condition": {
        "StringEquals": {
            "aws:SourceAccount": [
                "123456789012",
                "111122223333"
            ]
        },
        "ArnLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:us-east-1:123456789012:log-group:*",
                    "arn:aws:logs:us-east-1:111122223333:log-group:*"
                ]
```

```
            }
        }
    },
    {
        "Sid": "AllowPutObject",
        "Action": "s3:PutObject",
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
        "Condition": {
          "StringEquals": {
              "s3:x-amz-acl": "bucket-owner-full-control",
              "aws:SourceAccount": [
                  "123456789012",
                  "111122223333"
              ]
          },
          "ArnLike": {
                  "aws:SourceArn": [
                      "arn:aws:logs:us-east-1:123456789012:log-group:*",
                      "arn:aws:logs:us-east-1:111122223333:log-group:*"
                  ]
          }
        }
    }
]
}
```

2. 使用 put-bucket-policy 命令将您刚添加的策略设置为存储桶上的访问策略。此策略允许 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 amzn-s3-demo-bucket --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）将使用 SSE-KMS (CLI) 加密的 Amazon S3 存储桶导出到目标

仅当您要导出到使用服务器端加密的 S3 存储桶时，才需要执行此过程。 AWS KMS keys这种加密被称为 SSE-KMS。

导出到使用加密的存储桶 SSE-KMS

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：

   - *Region*替换为日志所在区域。
   - *account-ARN*替换为拥有 KMS 密钥的账户的 ARN。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow CWL Service Principal usage",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.Region.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "account-ARN"
            },
            "Action": [
                "kms:GetKeyPolicy*",
                "kms:PutKeyPolicy*",
                "kms:DescribeKey*",
                "kms:CreateAlias*",
```

```
                "kms:ScheduleKeyDeletion*",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

2. 输入以下命令：

```
aws kms create-key --policy file://key_policy.json
```

下面是此命令的示例输出：

```
{
    "KeyMetadata": {
        "AWSAccountId": "account_id",
        "KeyId": "key_id",
        "Arn": "arn:aws:kms:us-east-2:account-ARN:key/key_id",
        "CreationDate": "time",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
```

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，*amzn-s3-demo-bucket*替换为要将日志导出到的存储桶的名称。

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-
encryption-configuration file://bucketencryption.json
```

如果该命令没有返回错误，则该过程成功。

创建导出任务 (CLI)

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 设置访问权限 (CLI) 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 "amzn-s3-demo-bucket" --destination-prefix "export-
task-output"
```

下面是示例输出。

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

Cross-account 导出 (CLI)

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

- 创建用于跨账户导出的 S3 存储桶 (CLI)

- 设置跨账户导出 (CLI) 的访问权限
- 在 S3 存储桶上设置跨账户导出权限 (CLI)
- （可选）导出到目标 Amazon S3 存储桶加密后 SSE-KMS 用于跨账户导出 (CLI)
- 为跨账户导出 (CLI) 创建导出任务

创建用于跨账户导出的 S3 存储桶 (CLI)

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 桶必须与要导出的日志数据位于同一区域。 CloudWatch 日志不支持将数据导出到其他区域的 S3 存储桶。

要创建 S3 存储桶，请使用 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 amzn-s3-demo-bucket --create-bucket-configuration
 LocationConstraint=us-east-2
```

下面是示例输出。

```
{
    "Location": "/amzn-s3-demo-bucket"
}
```

设置跨账户导出 (CLI) 的访问权限

首先，您必须创建新的 IAM 策略，以使 CloudWatch Logs 能够在目标账户中对目标 Amazon S3 存储桶`s3:PutObject`执行操作。

除了`s3:PutObject`操作外，策略中包含的其他操作还取决于目标存储桶是使用 AWS KMS 加密还是使用 S3 对象所有权设置启用 ACL。

- 如果使用 KMS 加密，请为密钥资源添加 `kms:GenerateDataKey` 和 `kms:Decrypt` 操作

- 如果在存储桶上启用了 ACL，请为存储桶资源添加 `s3:PutObjectAcl` 操作

在以下策略中更改`amzn-s3-demo-bucket`目标 S3 存储桶的名称。

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。

JSON

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

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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
```

```
    ]
}
```

如果在目标存储桶上启用了 ACL，则在上述策略中将 s3: PutObjectAcl 添加到s3: PutObject 操作块。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "s3:PutObject",
               "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

要创建导出任务，您必须使用附加了AmazonS3ReadOnlyAccess托管策略、上面创建的 IAM 策略的 IAM 角色以及以下权限登录：

- 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。
- 通过身份提供者在 IAM 中托管的用户：

创建适用于身份联合验证的角色。按照《IAM 用户指南》中针对第三方身份提供者创建角色（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。

在 S3 存储桶上设置跨账户导出权限 (CLI)

默认情况下，所有 S3 桶和对象都是私有的。仅资源所有者（创建了存储桶的账户）能够访问存储桶及其包含的任何对象。不过，资源所有者可以选择通过编写访问策略来向其他资源和用户授予访问权限。

> **Important**
>
> 为了使导出到 S3 桶更加安全，我们现在要求您指定允许将日志数据导出到 S3 桶的源账户列表。
> 在以下示例中，aws:SourceAccount 密钥中账户 ID 的列表会是用户可以从其中将日志数据导出到 S3 存储桶的账户。aws:SourceArn 密钥是正在进行的操作的资源。如本示例所示，您可以将其限制为特定的日志组，也可以使用通配符。
> 建议您还包括创建 S3 桶的账户的账户 ID，以允许在同一账户内导出。

在 S3 桶上设置权限

1. 创建名为的文件，policy.json并将以下访问策略（更改amzn-s3-demo-bucket为目标 S3 存储桶的名称）添加到Principal要导出日志数据的区域的终端节点，例如us-west-1。使用文本编辑器以创建此策略文件。请勿使用 IAM 控制台。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
```

```
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
        "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
        "Condition": {
          "StringEquals": {
              "aws:SourceAccount": [
              "123456789012",
              "111122223333"
              ]
          },
          "ArnLike": {
                  "aws:SourceArn": [
                  "arn:aws:logs:us-east-1:123456789012:log-group:*",
                  "arn:aws:logs:us-east-1:111122223333:log-group:*"
                   ]
          }
        }
    },
    {
        "Action": "s3:PutObject" ,
        "Effect": "Allow",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
        "Condition": {
          "StringEquals": {
              "s3:x-amz-acl": "bucket-owner-full-control",
              "aws:SourceAccount": [
              "123456789012",
              "111122223333"
              ]
          },
          "ArnLike": {
                  "aws:SourceArn": [
                  "arn:aws:logs:us-east-1:123456789012:log-group:*",
                  "arn:aws:logs:us-east-1:111122223333:log-group:*"
                  ]
          }
        }
    },
    {
        "Effect": "Allow",
        "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/role_name"
        },
        "Action": "s3:PutObject",
```

```
            "Resource": "arn:aws:s3:::>amzn-s3-demo-bucket/*",
            "Condition": {
              "StringEquals": {
                  "s3:x-amz-acl": "bucket-owner-full-control"
              }
            }
        }
    ]
}
```

2. 使用 put-bucket-policy 命令将您刚添加的策略设置为存储桶上的访问策略。此策略允许 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 amzn-s3-demo-bucket --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）导出到目标 Amazon S3 存储桶加密后 SSE-KMS 用于跨账户导出 (CLI)

仅当您要导出到使用服务器端加密的 S3 存储桶时，才需要执行此过程。 AWS KMS keys这种加密被称为 SSE-KMS。

导出到使用加密的存储桶 SSE-KMS

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：

   - *us-east-1*替换为日志所在区域。
   - *account-ARN*替换为拥有 KMS 密钥的账户的 ARN。
   - *123456789012*替换为拥有 KMS 密钥的账号。
   - *key_id*使用 kms-key ID。
   - *role_name*使用用于创建导出任务的角色。

JSON

```
{
"Version":"2012-10-17",
"Statement": [
    {
        "Sid": "AllowCWLServicePrincipalUsage",
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.us-east-1.amazonaws.com"
        },
        "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
        ],
        "Resource": "*"
    },
    {
        "Sid": "EnableIAMUserPermissions",
        "Effect": "Allow",
        "Principal": {
            "AWS": "account-ARN"
        },
        "Action": [
            "kms:GetKeyPolicy*",
            "kms:PutKeyPolicy*",
            "kms:DescribeKey*",
            "kms:CreateAlias*",
            "kms:ScheduleKeyDeletion*",
            "kms:Decrypt"
        ],
        "Resource": "*"
    },
    {
        "Sid": "EnableIAMRolePermissions",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/role_name"
        },
        "Action": [
```

```
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

2. 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-1:123456789012:key/key_id",
        "CreationDate": "time",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
```

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，*amzn-s3-demo-bucket*替换为要将日志导出到的存储桶的名称。

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-
encryption-configuration file://bucketencryption.json
```

如果该命令没有返回错误，则该过程成功。

为跨账户导出 (CLI) 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 设置访问权限 (CLI) 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 "amzn-s3-demo-bucket" --destination-prefix "export-
task-output"
```

下面是示例输出。

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

描述导出任务 (CLI)

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": "amzn-s3-demo-bucket",
      "destinationPrefix": "export-task-output",
      "executionInfo": {
         "completionTime": 1441498600000
         "creationTime": 1441495400000
      },
      "from": 1441490400000,
      "logGroupName": "my-log-group",
      "status": {
         "code": "FAILED",
         "message": "FAILED"
      },
      "taskId": "cda45419-90ea-4db5-9833-aade86253e66",
      "taskName": "my-log-group-09-10-2015",
      "to": 1441494000000
   }]
}
```

取消导出任务 (CLI)

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

您可以在 Amazon Logs 中配置 CloudWatch 日志组，这样您就可以近乎实时地将数据流式传输到您的 Amazon S OpenSearch ervice 集群。有关更多信息，请参阅 Real-time 使用订阅处理日志数据。

> **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。

根据流式传输的日志数据量，请考虑设置函数级并发限制。有关更多信息，请参阅 Lambda 函数扩展。

> **Note**
>
> 由于将大量 CloudWatch 日志数据流式传输到 OpenSearch 服务可能会导致高额使用费，因此我们建议您在 AWS 账单与成本管理 控制台中创建预算。有关更多信息，请参阅使用 AWS 预算管理成本。

本节介绍在将日志组订阅到 S OpenSearch ervice 之前必须完成的先决条件。它还描述了如何为日志组订阅 OpenSearch 服务。

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：

     JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowOpenSearchStreamingAccess",
            "Action": [
                "es:*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:es:us-
east-1:123456789012:domain/cloudwatch-logs/*"
        }
    ]
}
```

   - 如果目标 OpenSearch 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 CloudWatch 日志的代码示例 AWS 软件开发工具包

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的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务服务结合来完成特定任务的代码示例。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 L CloudWatch ogs 与 AWS SDK。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 CloudWatch 日志的基本示例 AWS 软件开发工具包
  - 你好 CloudWatch 日志
  - 使用对 CloudWatch 日志执行的操作 AWS 软件开发工具包
    - AssociateKmsKey搭配使用 AWS SDK
    - CancelExportTask搭配使用 AWS SDK
    - CreateExportTask搭配使用 AWS SDK
    - CreateLogGroup搭配使用 AWS SDK 或 CLI
    - CreateLogStream搭配使用 AWS SDK 或 CLI
    - DeleteLogGroup搭配使用 AWS SDK 或 CLI
    - DeleteSubscriptionFilter搭配使用 AWS SDK
    - DeleteSyslogConfiguration搭配使用 AWS SDK
    - DescribeExportTasks搭配使用 AWS SDK
    - DescribeLogGroups搭配使用 AWS SDK 或 CLI
    - DescribeLogStreams搭配使用 AWS SDK 或 CLI
    - DescribeSubscriptionFilters搭配使用 AWS SDK
    - GetLogEvents搭配使用 AWS SDK 或 CLI
    - GetQueryResults搭配使用 AWS SDK
    - ListSyslogConfigurations搭配使用 AWS SDK
    - PutResourcePolicy搭配使用 AWS SDK

    - PutSubscriptionFilter搭配使用 AWS SDK
    - PutSyslogConfiguration搭配使用 AWS SDK
    - StartLiveTail搭配使用 AWS SDK 或 CLI
    - StartQuery搭配使用 AWS SDK
- 使用 CloudWatch 日志的场景 AWS 软件开发工具包
    - 配置容器服务连接
    - 创建您的第一个无服务器函数
    - 使用学习 CloudWatch 日志 syslog 提取基础知识 AWS SDK
    - 使用 CloudWatch 日志运行大型查询
    - 使用计划的事件调用 Lambda 函数

使用 CloudWatch 日志的基本示例 AWS 软件开发工具包

以下代码示例展示了如何在 AWS 软件开发工具包中使用 Amazon CloudWatch Logs 的基础知识。

示例

- 你好 CloudWatch 日志
- 使用对 CloudWatch 日志执行的操作 AWS 软件开发工具包
    - AssociateKmsKey搭配使用 AWS SDK
    - CancelExportTask搭配使用 AWS SDK
    - CreateExportTask搭配使用 AWS SDK
    - CreateLogGroup搭配使用 AWS SDK 或 CLI
    - CreateLogStream搭配使用 AWS SDK 或 CLI
    - DeleteLogGroup搭配使用 AWS SDK 或 CLI
    - DeleteSubscriptionFilter搭配使用 AWS SDK
    - DeleteSyslogConfiguration搭配使用 AWS SDK
    - DescribeExportTasks搭配使用 AWS SDK
    - DescribeLogGroups搭配使用 AWS SDK 或 CLI
    - DescribeLogStreams搭配使用 AWS SDK 或 CLI
    - DescribeSubscriptionFilters搭配使用 AWS SDK
    - GetLogEvents搭配使用 AWS SDK 或 CLI

- GetQueryResults搭配使用 AWS SDK
- ListSyslogConfigurations搭配使用 AWS SDK
- PutResourcePolicy搭配使用 AWS SDK
- PutSubscriptionFilter搭配使用 AWS SDK
- PutSyslogConfiguration搭配使用 AWS SDK
- StartLiveTail搭配使用 AWS SDK 或 CLI
- StartQuery搭配使用 AWS SDK

你好 CloudWatch 日志

以下代码示例显示了如何开始使用 L CloudWatch ogs。

Python

适用于 Python 的 SDK（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 hello_cloudwatch_logs() -> None:
    """
    Verifies connectivity to CloudWatch Logs by calling DescribeLogGroups
    and displaying the first page of log groups with name, ARN, and
    creation time.
    """
    logs_client = boto3.client("logs")

    try:
        response = logs_client.describe_log_groups()
        log_groups = response.get("logGroups", list())

        if not log_groups:
            print("No log groups found in this Region for your account.")
        else:
            print(f"Found {len(log_groups)} log group(s):\n")
            for lg in log_groups:
```

```
                name = lg.get("logGroupName", "N/A")
                arn = lg.get("arn", "N/A")
                creation_ms = lg.get("creationTime", 0)
                creation_dt = datetime.fromtimestamp(
                    creation_ms / 1000, tz=timezone.utc
                )
                print(f"  Name: {name}")
                print(f"  ARN:  {arn}")
                print(f"  Created: {creation_dt.strftime('%Y-%m-%d %H:%M:%S
UTC')}")
                print()

    except ClientError as error:
        logger.error(
            "Failed to describe log groups: %s",
            error.response["Error"]["Message"],
        )
        raise
```

- 有关 API 的详细信息，请参阅适用DescribeLogGroups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，请参阅将 L CloudWatch ogs 与 AWS SDK。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。

使用对 CloudWatch 日志执行的操作 AWS 软件开发工具包

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。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。

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。您可以在使用 CloudWatch 日志的场景 AWS 软件开发工具包中结合上下文查看操作。

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搭配使用 AWS SDK
- CancelExportTask搭配使用 AWS SDK

- CreateExportTask搭配使用 AWS SDK
- CreateLogGroup搭配使用 AWS SDK 或 CLI
- CreateLogStream搭配使用 AWS SDK 或 CLI
- DeleteLogGroup搭配使用 AWS SDK 或 CLI
- DeleteSubscriptionFilter搭配使用 AWS SDK
- DeleteSyslogConfiguration搭配使用 AWS SDK
- DescribeExportTasks搭配使用 AWS SDK
- DescribeLogGroups搭配使用 AWS SDK 或 CLI
- DescribeLogStreams搭配使用 AWS SDK 或 CLI
- DescribeSubscriptionFilters搭配使用 AWS SDK
- GetLogEvents搭配使用 AWS SDK 或 CLI
- GetQueryResults搭配使用 AWS SDK
- ListSyslogConfigurations搭配使用 AWS SDK
- PutResourcePolicy搭配使用 AWS SDK
- PutSubscriptionFilter搭配使用 AWS SDK
- PutSyslogConfiguration搭配使用 AWS SDK
- StartLiveTail搭配使用 AWS SDK 或 CLI
- StartQuery搭配使用 AWS SDK

`AssociateKmsKey`搭配使用 AWS SDK

以下代码示例演示了如何使用 `AssociateKmsKey`。

.NET

适用于 .NET 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。

```
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 适用于 .NET 的 AWS SDK 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，请参阅将 L CloudWatch ogs 与 AWS SDK。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`搭配使用 AWS SDK

以下代码示例演示了如何使用 `CancelExportTask`。

.NET

适用于 .NET 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。

```
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 适用于 .NET 的 AWS SDK 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，请参阅将 L CloudWatch ogs 与 AWS SDK。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`搭配使用 AWS SDK

以下代码示例演示了如何使用 `CreateExportTask`。

.NET

适用于 .NET 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。

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

```
   using Amazon.CloudWatchLogs.Model;

   /// <summary>
   /// Shows how to create an Export Task to export the contents of the Amazon
   /// CloudWatch Logs to the specified Amazon Simple Storage Service (Amazon
S3)
   /// bucket.
   /// </summary>
   public class CreateExportTask
   {
       public static async Task Main()
       {
           // This client object will be associated with the same AWS Region
           // as the default user on this system. If you need to use a
           // different AWS Region, pass it as a parameter to the client
           // constructor.
           var client = new AmazonCloudWatchLogsClient();
           string taskName = "export-task-example";
           string logGroupName = "cloudwatchlogs-example-loggroup";
           string destination = "amzn-s3-demo-bucket";
           var fromTime = 1437584472382;
           var toTime = 1437584472833;

           var request = new CreateExportTaskRequest
           {
               From = fromTime,
               To = toTime,
               TaskName = taskName,
               LogGroupName = logGroupName,
               Destination = destination,
           };

           var response = await client.CreateExportTaskAsync(request);

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK 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，请参阅将 L CloudWatch ogs 与 AWS SDK。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`搭配使用 AWS SDK 或 CLI

以下代码示例显示了如何使用`CreateLogGroup`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 配置容器服务连接
- 学习 syslog 提取基础知识

.NET

适用于 .NET 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。

```
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 适用于 .NET 的 AWS SDK API 参考CreateLogGroup中的。

CLI

AWS CLI

以下命令创建名为 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 适用于 JavaScript 的 AWS SDK 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，请参阅将 L CloudWatch ogs 与 AWS SDK。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`搭配使用 AWS SDK 或 CLI

以下代码示例显示了如何使用`CreateLogStream`。

.NET

适用于 .NET 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。

```
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 适用于 .NET 的 AWS SDK API 参考CreateLogStream中的。

CLI

AWS CLI

以下命令在日志组 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，请参阅将 L CloudWatch ogs 与 AWS SDK。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搭配使用 AWS SDK 或 CLI

以下代码示例显示了如何使用DeleteLogGroup。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 配置容器服务连接
- 创建您的第一个无服务器函数
- 学习 syslog 提取基础知识

.NET

适用于 .NET 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。

```
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 适用于 .NET 的 AWS SDK API 参考DeleteLogGroup中的。

CLI

AWS CLI

以下命令删除名为 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 适用于 JavaScript 的 AWS SDK 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，请参阅将 L CloudWatch ogs 与 AWS SDK。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`搭配使用 AWS SDK

以下代码示例显示了如何使用`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 适用于 C++ 的 AWS SDK API 参考DeleteSubscriptionFilter中的。

Java

适用于 Java 的 SDK 2.x

> **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 适用于 JavaScript 的 AWS SDK 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《适用于 JavaScript 的 AWS SDK 开发人员指南》https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/cloudwatch-examples-subscriptions.html#cloudwatch-examples-subscriptions-deleting。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK 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.fromEnvironment { 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，请参阅将 L CloudWatch ogs 与 AWS SDK。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。

`DeleteSyslogConfiguration`搭配使用 AWS SDK

以下代码示例演示了如何使用 `DeleteSyslogConfiguration`。

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：

- 学习 syslog 提取基础知识

Python

适用于 Python 的 SDK（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 delete_syslog_configuration(
        self, log_group_identifier: str, vpc_endpoint_id: str
    ) -> None:
        """
        Deletes a syslog configuration for a log group. After deletion, syslog
        data is no longer ingested through the specified VPC endpoint.

        :param log_group_identifier: The name or ARN of the log group.
        :param vpc_endpoint_id: The ID of the VPC endpoint associated with the
            syslog configuration.
        :raises ClientError: If the syslog configuration does not exist (in which
            case the error is logged but not re-raised during cleanup).
        """
        try:
            self.logs_client.delete_syslog_configuration(
                logGroupIdentifier=log_group_identifier,
                vpcEndpointId=vpc_endpoint_id,
            )
            logger.info(
                "Deleted syslog configuration for log group '%s' and VPC endpoint
'%s'.",
                log_group_identifier,
                vpc_endpoint_id,
            )
        except ClientError as error:
            if error.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info(
                    "Syslog configuration does not exist or was already deleted:
%s",
                    error.response["Error"]["Message"],
                )
            else:
```

```
                logger.error(
                    "Failed to delete syslog configuration: %s",
                    error.response["Error"]["Message"],
                )
                raise
```

- 有关 API 的详细信息，请参阅适用DeleteSyslogConfiguration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，请参阅将 L CloudWatch ogs 与 AWS SDK。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`搭配使用 AWS SDK

以下代码示例演示了如何使用 `DescribeExportTasks`。

.NET

适用于 .NET 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。

```
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 适用于 .NET 的 AWS SDK 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，请参阅将 L CloudWatch ogs 与 AWS SDK。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`搭配使用 AWS SDK 或 CLI

以下代码示例显示了如何使用`DescribeLogGroups`。

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：

- 学习 syslog 提取基础知识

.NET

适用于 .NET 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。

```
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 适用于 .NET 的 AWS SDK API 参考[DescribeLogGroups](#)中的。

CLI

AWS CLI

以下命令描述名为 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?.every((lg) => !!lg)) {
      logGroups.push(...page.logGroups);
```

```
    }
  }

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

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK 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，请参阅将 L CloudWatch ogs 与 AWS SDK。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。

DescribeLogStreams搭配使用 AWS SDK 或 CLI

以下代码示例显示了如何使用DescribeLogStreams。

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：

- 创建您的第一个无服务器函数

CLI

AWS CLI

以下命令显示日志组 my-logs 中以前缀 2015 开头的所有日志流：

```
aws logs describe-log-streams --log-group-name my-logs --log-stream-name-
prefix 2015
```

输出：

```
{
    "logStreams": [
        {
            "creationTime": 1433189871774,
            "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-
stream:20150531",
            "logStreamName": "20150531",
```

```
            "storedBytes": 0
        },
        {
            "creationTime": 1433189873898,
            "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-
stream:20150601",
            "logStreamName": "20150601",
            "storedBytes": 0
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeLogStreams中的。

Java

适用于 Java 的 SDK 2.x

> **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。

在指定日志组中搜索与给定前缀匹配的日志流。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CloudWatchLogsSearch {

    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                <logGroupName> <logStreamName>

            Where:
                logGroupName - The name of the log group (for example,
WeathertopJavaContainerLogs).
                logStreamName - The name of the log stream (for example,
weathertop-java-stream).
                pattern - the pattern to use (for example, INFO)

            """;

       if (args.length != 3) {
           System.out.print(usage);
           System.exit(1);
       }

       String logGroupName = args[0] ;
       String logStreamName = args[1] ;
       String pattern = args[2] ;

       CloudWatchLogsClient cwlClient = CloudWatchLogsClient.builder()
               .region(Region.US_EAST_1)
               .build();

       searchLogStreamsAndFilterEvents(cwlClient, logGroupName, logStreamName,
pattern);
   }

   /**
    * Searches for log streams with a specific prefix within a log group and
filters log events based on a specified pattern.
    *
    * @param cwlClient       the CloudWatchLogsClient used to interact with AWS
CloudWatch Logs
    * @param logGroupName    the name of the log group to search within
    * @param logStreamPrefix the prefix of the log streams to search for
    * @param pattern         the pattern to filter log events by
    */
   public static void searchLogStreamsAndFilterEvents(CloudWatchLogsClient
cwlClient, String logGroupName, String logStreamPrefix, String pattern) {
       DescribeLogStreamsRequest describeLogStreamsRequest =
DescribeLogStreamsRequest.builder()
               .logGroupName(logGroupName)
               .logStreamNamePrefix(logStreamPrefix)
```

```
                .build();

        DescribeLogStreamsResponse describeLogStreamsResponse =
 cwlClient.describeLogStreams(describeLogStreamsRequest);
        List<LogStream> logStreams = describeLogStreamsResponse.logStreams();

        for (LogStream logStream : logStreams) {
            String logStreamName = logStream.logStreamName();
            System.out.println("Searching in log stream: " + logStreamName);

            FilterLogEventsRequest filterLogEventsRequest =
 FilterLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamNames(logStreamName)
                    .filterPattern(pattern)
                    .build();

            FilterLogEventsResponse filterLogEventsResponse =
 cwlClient.filterLogEvents(filterLogEventsRequest);

            for (FilteredLogEvent event : filterLogEventsResponse.events()) {
                System.out.println(event.message());
            }


 System.out.println("-------------------------------------------------"); //
 Separator for better readability
        }
    }
}
```

打印有关指定日志组中最新日志流的元数据。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class CloudWatchLogQuery {
    public static void main(final String[] args) {
        final String usage = """
                Usage:
                  <logGroupName>

                Where:
                  logGroupName - The name of the log group (for example, /aws/
lambda/ChatAIHandler).
                """;

        if (args.length != 1) {
            System.out.print(usage);
            System.exit(1);
        }

        String logGroupName = "/aws/lambda/ChatAIHandler" ; //args[0];
        CloudWatchLogsClient logsClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeMostRecentLogStream(logsClient, logGroupName);
    }

    /**
     * Describes and prints metadata about the most recent log stream in the
 specified log group.
     *
     * @param logsClient   the CloudWatchLogsClient used to interact with AWS
 CloudWatch Logs
     * @param logGroupName the name of the log group
     */
    public static void describeMostRecentLogStream(CloudWatchLogsClient
 logsClient, String logGroupName) {
        DescribeLogStreamsRequest streamsRequest =
 DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .orderBy(OrderBy.LAST_EVENT_TIME)
                .descending(true)
                .limit(1)
                .build();

        try {
```

```
            DescribeLogStreamsResponse streamsResponse =
 logsClient.describeLogStreams(streamsRequest);
            List<LogStream> logStreams = streamsResponse.logStreams();

            if (logStreams.isEmpty()) {
                System.out.println("No log streams found for log group: " +
 logGroupName);
                return;
            }

            LogStream stream = logStreams.get(0);
            System.out.println("Most Recent Log Stream:");
            System.out.println("  Name: " + stream.logStreamName());
            System.out.println("  ARN: " + stream.arn());
            System.out.println("  Creation Time: " + stream.creationTime());
            System.out.println("  First Event Time: " +
 stream.firstEventTimestamp());
            System.out.println("  Last Event Time: " +
 stream.lastEventTimestamp());
            System.out.println("  Stored Bytes: " + stream.storedBytes());
            System.out.println("  Upload Sequence Token: " +
 stream.uploadSequenceToken());

        } catch (CloudWatchLogsException e) {
            System.err.println("Failed to describe log stream: " +
 e.awsErrorDetails().errorMessage());
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeLogStreams中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 L CloudWatch ogs 与 AWS SDK。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`搭配使用 AWS SDK

以下代码示例显示了如何使用`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 适用于 C++ 的 AWS SDK API 参考DescribeSubscriptionFilters中的。

Java

适用于 Java 的 SDK 2.x

> **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 适用于 JavaScript 的 AWS SDK 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《适用于 JavaScript 的 AWS SDK 开发人员指南》https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/cloudwatch-examples-subscriptions.html#cloudwatch-examples-subscriptions-describing。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK 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.fromEnvironment { region = "us-west-2" }.use { cwlClient
 ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name  ${filter.filterName} pattern
 ${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- 有关 API 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，请参阅将 L CloudWatch ogs 与 AWS SDK。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。

`GetLogEvents`搭配使用 AWS SDK 或 CLI

以下代码示例显示了如何使用`GetLogEvents`。

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：

- 创建您的第一个无服务器函数

CLI

AWS CLI

以下命令从日志组 `my-logs` 中名为 `20150601` 的日志流中检索日志事件：

```
aws logs get-log-events --log-group-name my-logs --log-stream-name 20150601
```

输出：

```
{
    "nextForwardToken":
 "f/31961209122447488583055879464742346735121166569214640130",
    "events": [
        {
            "ingestionTime": 1433190494190,
            "timestamp": 1433190184356,
            "message": "Example Event 1"
        },
        {
            "ingestionTime": 1433190516679,
            "timestamp": 1433190184356,
            "message": "Example Event 1"
        },
        {
            "ingestionTime": 1433190494190,
            "timestamp": 1433190184358,
            "message": "Example Event 2"
        }
    ],
    "nextBackwardToken":
 "b/31961209122358285602261756944988674324553373268216709120"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetLogEvents](#)中的。

Java

适用于 Java 的 SDK 2.x

> **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.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
```

```
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsResponse;

import java.time.Instant;
import java.time.temporal.ChronoUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetLogEvents {

    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
                  logGroupName - The name of the log group (for example,
 myloggroup).
                  logStreamName - The name of the log stream (for example,
 mystream).

                """;

       // if (args.length != 2) {
       //     System.out.print(usage);
       //     System.exit(1);
//        }

        String logGroupName = "WeathertopJavaContainerLogs" ; //args[0];
        String logStreamName = "weathertop-java-stream" ; //args[1];

        Region region = Region.US_EAST_1 ;
```

```
        CloudWatchLogsClient cloudWatchLogsClient =
CloudWatchLogsClient.builder()
                .region(region)
                .build();

        getCWLogEvents(cloudWatchLogsClient, logGroupName, logStreamName);
        cloudWatchLogsClient.close();
    }

    public static void getCWLogEvents(CloudWatchLogsClient cloudWatchLogsClient,
                                      String logGroupName,
                                      String logStreamPrefix) {
        try {
            // First, find the exact log stream name
            DescribeLogStreamsRequest describeRequest =
DescribeLogStreamsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamNamePrefix(logStreamPrefix)
                    .limit(1) // get the first matching stream
                    .build();

            DescribeLogStreamsResponse describeResponse =
cloudWatchLogsClient.describeLogStreams(describeRequest);

            if (describeResponse.logStreams().isEmpty()) {
                System.out.println("No matching log streams found for prefix: " +
logStreamPrefix);
                return;
            }

            String exactLogStreamName =
describeResponse.logStreams().get(0).logStreamName();
            System.out.println("Using exact log stream: " + exactLogStreamName);

            long startTime = Instant.now().minus(7,
ChronoUnit.DAYS).toEpochMilli();
            long endTime = Instant.now().toEpochMilli();

            GetLogEventsRequest getLogEventsRequest =
GetLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamName(exactLogStreamName) // <-- exact name, not
prefix
                    .startTime(startTime)
```

```
                    .endTime(endTime)
                    .startFromHead(true)
                    .build();

            GetLogEventsResponse response =
 cloudWatchLogsClient.getLogEvents(getLogEventsRequest);

            if (response.events().isEmpty()) {
                System.out.println("No log events found in the past 7 days.");
            } else {
                response.events().forEach(e -> System.out.println(e.message()));
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetLogEvents中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 L CloudWatch ogs 与 AWS SDK。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`搭配使用 AWS SDK

以下代码示例显示了如何使用`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

.NET

适用于 .NET 的 SDK (v4)

> **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。

```
    /// <summary>
    /// Gets the results of a CloudWatch Logs Insights query.
    /// </summary>
    /// <param name="queryId">The ID of the query.</param>
    /// <returns>The query results response.</returns>
    public async Task<GetQueryResultsResponse?> GetQueryResultsAsync(string
queryId)
    {
        try
        {
            var request = new GetQueryResultsRequest
            {
                QueryId = queryId
            };

            var response = await
_amazonCloudWatchLogs.GetQueryResultsAsync(request);
            return response;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError($"Query not found: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while getting query results:
{ex.Message}");
            return null;
        }
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考GetQueryResults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。

```
/**
 * 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 适用于 JavaScript 的 AWS SDK API 参考GetQueryResults中的。

Python

适用于 Python 的 SDK（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。

SAP ABAP

适用于 SAP ABAP 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。

```
TRY.
    oo_result = lo_cwl->getqueryresults(
      iv_queryid = iv_query_id ).

    " Display query status and result count
    DATA(lv_status) = oo_result->get_status( ).
    DATA(lt_results) = oo_result->get_results( ).
    DATA(lv_result_count) = lines( lt_results ).
```

```
        MESSAGE |Query status: { lv_status }. Retrieved { lv_result_count } log
event(s).| TYPE 'I'.
      CATCH /aws1/cx_cwlinvalidparameterex.
        MESSAGE 'Invalid parameter.' TYPE 'E'.
      CATCH /aws1/cx_cwlresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'E'.
      CATCH /aws1/cx_cwlserviceunavailex.
        MESSAGE 'Service unavailable.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 的详细信息，请参阅适用GetQueryResults于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 L CloudWatch ogs 与 AWS SDK。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。

`ListSyslogConfigurations`搭配使用 AWS SDK

以下代码示例演示了如何使用 `ListSyslogConfigurations`。

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：

- 学习 syslog 提取基础知识

Python

适用于 Python 的 SDK（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 list_syslog_configurations(
        self,
        log_group_identifier: Optional[str] = None,
        vpc_endpoint_id: Optional[str] = None,
    ) -> list[dict[str, Any]]:
        """
```

```
        Lists syslog configurations, optionally filtered by log group or VPC
endpoint.

        Handles pagination by following nextToken until all results are returned.

        :param log_group_identifier: Optional log group name or ARN to filter by.
        :param vpc_endpoint_id: Optional VPC endpoint ID to filter by.
        :return: A list of syslog configuration dictionaries.
        :raises ClientError: If a filter parameter is invalid.
        """
        try:
            configurations = list()
            params = dict()
            if log_group_identifier is not None:
                params["logGroupIdentifier"] = log_group_identifier
            if vpc_endpoint_id is not None:
                params["vpcEndpointId"] = vpc_endpoint_id

            while True:
                response = self.logs_client.list_syslog_configurations(**params)
                configurations.extend(response.get("syslogConfigurations",
list()))
                next_token = response.get("nextToken", None)
                if next_token is None:
                    break
                params["nextToken"] = next_token

            logger.info("Listed %d syslog configuration(s).",
len(configurations))
            return configurations
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidParameterException":
                logger.error(
                    "Invalid filter parameter: %s. Ensure the log group
identifier "
                    "or VPC endpoint ID is correctly formatted.",
                    error.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Failed to list syslog configurations: %s",
                    error.response["Error"]["Message"],
                )
            raise
```

- 有关 API 的详细信息，请参阅适用ListSyslogConfigurations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，请参阅将 L CloudWatch ogs 与 AWS SDK。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。

`PutResourcePolicy`搭配使用 AWS SDK

以下代码示例演示了如何使用 `PutResourcePolicy`。

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：

- 学习 syslog 提取基础知识

Python

适用于 Python 的 SDK（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 put_resource_policy(
    self,
    log_group_arn: str,
    vpc_endpoint_id: str,
) -> dict[str, Any]:
    """
    Creates or updates a resource-scoped policy on a log group that grants
    the syslog.logs.amazonaws.com service principal permission to call
    logs:PutLogEvents and logs:CreateLogStream.

    A resource-scoped policy (one attached to a specific log group) is
    specified with resourceArn and policyDocument. The PutResourcePolicy
```

```
        API does not allow policyName to be combined with resourceArn.

        :param log_group_arn: The ARN of the log group (with trailing :*).
        :param vpc_endpoint_id: The VPC endpoint ID to scope the condition.
        :return: The resource policy response.
        """
        policy_document = json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "SyslogIngestPermissions",
                        "Effect": "Allow",
                        "Principal": {"Service": "syslog.logs.amazonaws.com"},
                        "Action": [
                            "logs:PutLogEvents",
                            "logs:CreateLogStream",
                        ],
                        "Resource": log_group_arn,
                        "Condition": {
                            "StringEquals": {"aws:sourceVpce": vpc_endpoint_id}
                        },
                    }
                ],
            }
        )
        try:
            response = self.logs_client.put_resource_policy(
                policyDocument=policy_document,
                resourceArn=log_group_arn,
            )
            logger.info("Put resource policy on log group '%s'.", log_group_arn)
            return response
        except ClientError as error:
            logger.error(
                "Failed to put resource policy: %s",
                error.response["Error"]["Message"],
            )
            raise
```

- 有关 API 的详细信息，请参阅适用PutResourcePolicy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，请参阅将 L CloudWatch ogs 与 AWS SDK。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搭配使用 AWS SDK

以下代码示例显示了如何使用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 适用于 C++ 的 AWS SDK API 参考PutSubscriptionFilter中的。

Java

适用于 Java 的 SDK 2.x

> **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 适用于 JavaScript 的 AWS SDK 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《适用于 JavaScript 的 AWS SDK 开发人员指南》https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/cloudwatch-examples-subscriptions.html#cloudwatch-examples-subscriptions-creating。

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK 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，请参阅将 L CloudWatch ogs 与 AWS SDK。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。

`PutSyslogConfiguration`搭配使用 AWS SDK

以下代码示例演示了如何使用 `PutSyslogConfiguration`。

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：

- 学习 syslog 提取基础知识

Python

适用于 Python 的 SDK（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 put_syslog_configuration(
        self, log_group_identifier: str, vpc_endpoint_id: str
    ) -> None:
        """
        Creates or updates a syslog configuration for a log group. This enables
        ingestion of syslog data through the specified VPC endpoint.

        :param log_group_identifier: The name or ARN of the log group to
associate
            with the syslog configuration.
        :param vpc_endpoint_id: The ID of the VPC endpoint to use for syslog
            ingestion.
        :raises ClientError: If the log group or VPC endpoint does not exist.
        """
        try:
```

```
            self.logs_client.put_syslog_configuration(
                logGroupIdentifier=log_group_identifier,
                vpcEndpointId=vpc_endpoint_id,
            )
            logger.info(
                "Created syslog configuration for log group '%s' with VPC
endpoint '%s'.",
                log_group_identifier,
                vpc_endpoint_id,
            )
        except ClientError as error:
            if error.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "The specified log group or VPC endpoint does not exist: %s",
                    error.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Failed to put syslog configuration: %s",
                    error.response["Error"]["Message"],
                )
            raise
```

- 有关 API 的详细信息，请参阅适用PutSyslogConfiguration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，请参阅将 L CloudWatch ogs 与 AWS SDK。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`搭配使用 AWS SDK 或 CLI

以下代码示例显示了如何使用`StartLiveTail`。

.NET

适用于 .NET 的 SDK

包含所需的文件。

```
using Amazon;
```

```
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
```

启动 Live Tail 会话

```
            var client = new AmazonCloudWatchLogsClient();
            var request = new StartLiveTailRequest
            {
                LogGroupIdentifiers = logGroupIdentifiers,
                LogStreamNames = logStreamNames,
                LogEventFilterPattern = filterPattern,
            };

            var response = await client.StartLiveTailAsync(request);

            // Catch if request fails
            if (response.HttpStatusCode != System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("Failed to start live tail session");
                return;
            }
```

您可以通过两种方式处理 Live Tail 会话中的事件：

```
            /* Method 1
            * 1). Asynchronously loop through the event stream
            * 2). Set a timer to dispose the stream and stop the Live Tail
 session at the end.
            */
            var eventStream = response.ResponseStream;
            var task = Task.Run(() =>
            {
                foreach (var item in eventStream)
                {
                    if (item is LiveTailSessionUpdate liveTailSessionUpdate)
                    {
                        foreach (var sessionResult in
 liveTailSessionUpdate.SessionResults)
                        {
                            Console.WriteLine("Message : {0}",
 sessionResult.Message);
```

```
                    }
                }
                if (item is LiveTailSessionStart)
                {
                    Console.WriteLine("Live Tail session started");
                }
                // On-stream exceptions are processed here
                if (item is CloudWatchLogsEventStreamException)
                {
                    Console.WriteLine($"ERROR: {item}");
                }
            }
        });
        // Close the stream to stop the session after a timeout
        if (!task.Wait(TimeSpan.FromSeconds(10))){
            eventStream.Dispose();
            Console.WriteLine("End of line");
        }
```

```
        /* Method 2
        * 1). Add event handlers to each event variable
        * 2). Start processing the stream and wait for a timeout using
AutoResetEvent
        */
        AutoResetEvent endEvent = new AutoResetEvent(false);
        var eventStream = response.ResponseStream;
        using (eventStream) // automatically disposes the stream to stop the
session after execution finishes
        {
            eventStream.SessionStartReceived += (sender, e) =>
            {
                Console.WriteLine("LiveTail session started");
            };
            eventStream.SessionUpdateReceived += (sender, e) =>
            {
                foreach (LiveTailSessionLogEvent logEvent in
e.EventStreamEvent.SessionResults){
                    Console.WriteLine("Message: {0}", logEvent.Message);
                }
            };
            // On-stream exceptions are captured here
            eventStream.ExceptionReceived += (sender, e) =>
```

```
                {
                    Console.WriteLine($"ERROR:
{e.EventStreamException.Message}");
                };

                eventStream.StartProcessing();
                // Stream events for this amount of time.
                endEvent.WaitOne(TimeSpan.FromSeconds(10));
                Console.WriteLine("End of line");
            }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考StartLiveTail中的。

CLI

AWS CLI

以下命令在名为的日志组上启动 Live Tail 会话my-logs：

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-
east-1:111111222222:log-group:my-logs
```

以下命令在以交互模式命名的my-logs日志组上启动 Live Tail 会话：

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-
east-1:111111222222:log-group:my-logs --mode interactive
```

在交互模式下，您可以在尾部日志中突出显示多达五个术语。默认情况下，严重性代码会突出显示。按下h进入高亮显示模式，然后逐一键入要突出显示的术语，然后按 Enter。按c下可清除突出显示的术语。按t下可在 JSON/Plain 文本之间切换格式。按下Esc退出 Live Tail 会话。按up/down键在日志事件之间向上或向下滚动，使用Ctrl + u/Ctrl + d可以更快地滚动。按下q可滚动至最新的日志事件。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考StartLiveTail中的。

Go

适用于 Go 的 SDK V2

包含所需的文件。

```
import (
 "context"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)
```

处理 Live Tail 会话中的事件。

```
func handleEventStreamAsync(stream *cloudwatchlogs.StartLiveTailEventStream) {
 eventsChan := stream.Events()
 for {
  event := <-eventsChan
  switch e := event.(type) {
  case *types.StartLiveTailResponseStreamMemberSessionStart:
   log.Println("Received SessionStart event")
  case *types.StartLiveTailResponseStreamMemberSessionUpdate:
   for _, logEvent := range e.Value.SessionResults {
    log.Println(*logEvent.Message)
   }
  default:
   // Handle on-stream exceptions
   if err := stream.Err(); err != nil {
    log.Fatalf("Error 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 适用于 Go 的 AWS SDK API 参考StartLiveTail中的。

Java

适用于 Java 的 SDK 2.x

包含所需的文件。

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
```

```
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

处理 Live Tail 会话中的事件。

```
    private static StartLiveTailResponseHandler
 getStartLiveTailResponseStreamHandler(
            AtomicReference<Subscription> subscriptionAtomicReference) {
        return StartLiveTailResponseHandler.builder()
            .onResponse(r -> System.out.println("Received initial response"))
            .onError(throwable -> {
                CloudWatchLogsException e = (CloudWatchLogsException)
 throwable.getCause();
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            })
            .subscriber(() -> new FlowableSubscriber<>() {
                @Override
                public void onSubscribe(@NonNull Subscription s) {
                    subscriptionAtomicReference.set(s);
                    s.request(Long.MAX_VALUE);
                }

                @Override
                public void onNext(StartLiveTailResponseStream event) {
                    if (event instanceof LiveTailSessionStart) {
```

```
                        LiveTailSessionStart sessionStart =
(LiveTailSessionStart) event;
                        System.out.println(sessionStart);
                    } else if (event instanceof LiveTailSessionUpdate) {
                        LiveTailSessionUpdate sessionUpdate =
(LiveTailSessionUpdate) event;
                        List<LiveTailSessionLogEvent> logEvents =
sessionUpdate.sessionResults();
                        logEvents.forEach(e -> {
                            long timestamp = e.timestamp();
                            Date date = new Date(timestamp);
                            System.out.println("[" + date + "] " + e.message());
                        });
                    } else {
                        throw CloudWatchLogsException.builder().message("Unknown
event type").build();
                    }
                }

                @Override
                public void onError(Throwable throwable) {
                    System.out.println(throwable.getMessage());
                    System.exit(1);
                }

                @Override
                public void onComplete() {
                    System.out.println("Completed Streaming Session");
                }
            })
            .build();
    }
```

启动 Live Tail 会话。

```
        CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
                CloudWatchLogsAsyncClient.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();

        StartLiveTailRequest request =
                StartLiveTailRequest.builder()
```

```
                    .logGroupIdentifiers(logGroupIdentifiers)
                    .logStreamNames(logStreamNames)
                    .logEventFilterPattern(logEventFilterPattern)
                    .build();

        /* Create a reference to store the subscription */
        final AtomicReference<Subscription> subscriptionAtomicReference = new
AtomicReference<>(null);

        cloudWatchLogsAsyncClient.startLiveTail(request,
getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));
```

经过一段时间后停止 Live Tail 会话。

```
        /* Set a timeout for the session and cancel the subscription. This will:
         * 1). Close the stream
         * 2). Stop the Live Tail session
         */
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        if (subscriptionAtomicReference.get() != null) {
            subscriptionAtomicReference.get().cancel();
            System.out.println("Subscription to stream closed");
        }
```

- 有关 API 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 适用于 JavaScript 的 AWS SDK 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

适用于 Python 的 SDK（Boto3）

包含所需的文件。

```
import boto3
import time
from datetime import datetime
```

启动 Live Tail 会话

```
    # Initialize the client
    client = boto3.client('logs')

    start_time = time.time()
```

```
    try:
        response = client.start_live_tail(
            logGroupIdentifiers=log_group_identifiers,
            logStreamNames=log_streams,
            logEventFilterPattern=filter_pattern
        )
        event_stream = response['responseStream']
        # Handle the events streamed back in the response
        for event in event_stream:
            # Set a timeout to close the stream.
            # This will end the Live Tail session.
            if (time.time() - start_time >= 10):
                event_stream.close()
                break
            # Handle when session is started
            if 'sessionStart' in event:
                session_start_event = event['sessionStart']
                print(session_start_event)
            # Handle when log event is given in a session update
            elif 'sessionUpdate' in event:
                log_events = event['sessionUpdate']['sessionResults']
                for log_event in log_events:
                    print('[{date}]
{log}'.format(date=datetime.fromtimestamp(log_event['timestamp']/1000),log=log_event['me
            else:
                # On-stream exceptions are captured here
                raise RuntimeError(str(event))
    except Exception as e:
        print(e)
```

- 有关 API 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，请参阅将 L CloudWatch ogs 与 AWS SDK。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搭配使用 AWS SDK

以下代码示例显示了如何使用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

.NET

适用于 .NET 的 SDK (v4)

> **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。

```
    /// <summary>
    /// Starts a CloudWatch Logs Insights query.
    /// </summary>
    /// <param name="logGroupName">The name of the log group to query.</param>
    /// <param name="queryString">The CloudWatch Logs Insights query string.</
param>
    /// <param name="startTime">The start time for the query (seconds since
 epoch).</param>
    /// <param name="endTime">The end time for the query (seconds since epoch).</
param>
    /// <param name="limit">The maximum number of results to return.</param>
    /// <returns>The query ID if successful, null otherwise.</returns>
    public async Task<string?> StartQueryAsync(
        string logGroupName,
        string queryString,
        long startTime,
        long endTime,
        int limit = 10000)
    {
        try
        {
            var request = new StartQueryRequest
            {
                LogGroupName = logGroupName,
                QueryString = queryString,
                StartTime = startTime,
                EndTime = endTime,
                Limit = limit
            };
```

```
            var response = await _amazonCloudWatchLogs.StartQueryAsync(request);
            return response.QueryId;
        }
        catch (InvalidParameterException ex)
        {
            _logger.LogError($"Invalid parameter for query: {ex.Message}");
            return null;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError($"Log group not found: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while starting query:
{ex.Message}");
            return null;
        }
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考StartQuery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。

```
/**
 * 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 适用于 JavaScript 的 AWS SDK API 参考StartQuery中的。

Python

适用于 Python 的 SDK（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_group,
                    startTime=start_time,
                    endTime=end_time,
                    queryString=self.query_string,
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.
```

```
        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")
```

- 有关 API 的详细信息，请参阅适用StartQuery于 Python 的AWS SDK (Boto3) API 参考。

SAP ABAP

适用于 SAP ABAP 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。

```
   TRY.
       " iv_log_group_name = '/aws/lambda/my-function'
       " iv_query_string = 'fields @timestamp, @message | sort @timestamp desc |
limit 20'
       " iv_start_time and iv_end_time must be in Unix epoch milliseconds (ms
since Jan 1, 1970 00:00:00 UTC)
       oo_result = lo_cwl->startquery(
         iv_loggroupname = iv_log_group_name
         iv_starttime    = iv_start_time
         iv_endtime      = iv_end_time
         iv_querystring  = iv_query_string
         iv_limit        = iv_limit ).

       " Display the query ID for tracking
       DATA(lv_query_id) = oo_result->get_queryid( ).
       MESSAGE |Query started successfully with ID: { lv_query_id }| TYPE 'I'.
     CATCH /aws1/cx_cwlinvalidparameterex.
       MESSAGE 'Invalid parameter.' TYPE 'E'.
     CATCH /aws1/cx_cwllimitexceededex.
       MESSAGE 'Limit exceeded.' TYPE 'E'.
     CATCH /aws1/cx_cwlmalformedqueryex.
       MESSAGE 'Malformed query.' TYPE 'E'.
     CATCH /aws1/cx_cwlresourcenotfoundex.
       MESSAGE 'Resource not found.' TYPE 'E'.
     CATCH /aws1/cx_cwlserviceunavailex.
       MESSAGE 'Service unavailable.' TYPE 'E'.
   ENDTRY.
```

- 有关 API 的详细信息，请参阅适用StartQuery于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 L CloudWatch ogs 与 AWS SDK。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 CloudWatch 日志的场景 AWS 软件开发工具包

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 中的多个函数或与其他函数组合来完成特定任务 AWS 服务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- 配置容器服务连接
- 创建您的第一个无服务器函数
- 使用学习 CloudWatch 日志 syslog 提取基础知识 AWS SDK
- 使用 CloudWatch 日志运行大型查询
- 使用计划的事件调用 Lambda 函数

配置容器服务连接

以下代码示例展示了如何：

- 创建 VPC 基础设施
- 设置日志记录
- 创建 ECS 集群
- 配置 IAM 角色
- 使用 Service Connect 创建服务
- 验证部署
- 清理资源

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits
```

```
set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}
```

```
# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
```

```
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
```

```
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }
```

```
    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
```

```
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
```

```
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
```

```
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
```

```
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
```

```
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================="
    echo "CREATED RESOURCES"
    echo "============================================="
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================="
    echo ""
```

```
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
```

```
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
```

```
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources
```

```
    # Ask user if they want to clean up
    echo ""
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or
 the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
 dependency issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace

- DeleteRole
- DeleteSecurityGroup
- DeleteService (AWS Cloud Map)
- DeleteService （亚马逊 ECS ）
- DeregisterTaskDefinition
- DescribeClusters
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait （Amazon EC2 ）
- Wait （Amazon ECS ）

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 L CloudWatch ogs 与 AWS SDK。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。

创建您的第一个无服务器函数

以下代码示例展示了如何：

- 为 Lambda 创建 IAM 角色
- 创建函数代码
- 创建 Lambda 函数

- 测试 Lambda 函数
- 清理资源

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

################################################################################
# Setup
################################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1
```

```
declare -a CREATED_RESOURCES

################################################################################
# Helper functions
################################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"

        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
 2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function
 deletion."
                        break
                    fi
```

```
                done
                ;;
            iam-role-policy)
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could
 not detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
 IAM role ${NAME}."
                ;;
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "========================================="
    echo "ERROR: Script failed at $1"
    echo "========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
```

```
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

################################################################################
# Region pre-check
################################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
```

```
    exit 1
fi

################################################################################
# Runtime selection
################################################################################

echo ""
echo "==========================================="
echo "AWS Lambda - Create Your First Function"
echo "==========================================="
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in
 event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
```

```
        raise
def calculate_area(length, width):
    return length * width
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !==
 'number') {
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

################################################################################
# Step 1: Create IAM execution role
################################################################################

echo ""
echo "=========================================="
echo "Step 1: Create IAM execution role"
echo "=========================================="
echo ""

TRUST_POLICY='{
```

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

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi

echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/
service-role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
```

```
echo "Waiting for IAM role to propagate..."
sleep 10

################################################################################
# Step 2: Create Lambda function
################################################################################

echo ""
echo "=========================================="
echo "Step 2: Create Lambda function"
echo "=========================================="
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
```

```
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi

echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --
query State --output text" \
    "Active"

################################################################################
# Step 3: Invoke the function
################################################################################

echo ""
echo "=========================================="
echo "Step 3: Invoke the function"
echo "=========================================="
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi
```

```
RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

################################################################################
# Step 4: View CloudWatch logs
################################################################################

echo ""
echo "========================================="
echo "Step 4: View CloudWatch Logs"
echo "========================================="
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
```

```
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

################################################################################
# Summary and cleanup
################################################################################

echo ""
echo "==========================================="
echo "SUMMARY"
echo "==========================================="
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。

- AttachRolePolicy
- CreateFunction
- CreateRole
- DeleteFunction
- DeleteLogGroup
- DeleteRole
- DescribeLogStreams
- DetachRolePolicy
- GetFunction
- GetFunctionConfiguration
- GetLogEvents
- Invoke

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 L CloudWatch ogs 与 AWS SDK。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。

使用学习 CloudWatch 日志 syslog 提取基础知识 AWS SDK

以下代码示例展示了如何：

- 部署预置 syslog VPC 端点的 CloudFormation 堆栈。
- 创建日志组并为 syslog 服务添加资源策略。
- 使用创建系统日志配置。 PutSyslogConfiguration
- 列出带有和不带过滤器的系统日志配置。
- 通过删除 syslog 配置、日志组和堆栈进行清理。

Python

适用于 Python 的 SDK（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。

在命令提示符中运行交互式场景。

```
class SyslogIngestionScenario:
    """Interactive scenario demonstrating CloudWatch Logs syslog ingestion."""

    STACK_NAME = "syslog-demo-stack"

    def __init__(
        self,
        cloudwatch_logs_wrapper: CloudWatchLogsWrapper,
        cf_client: boto3.client,
    ) -> None:
        """
        :param cloudwatch_logs_wrapper: A CloudWatchLogsWrapper instance.
        :param cf_client: A Boto3 CloudFormation client.
        """
        self.wrapper = cloudwatch_logs_wrapper
        self.cf_client = cf_client
        self.vpc_endpoint_id: Optional[str] = None
        self.log_group_name: Optional[str] = None
        self.log_group_arn: Optional[str] = None
        self.stack_deployed = False

    # ------------------------------------------------------------------
    # Setup phase
    # ------------------------------------------------------------------
    def deploy_cfn_stack(self) -> str:
        """
        Deploys the CloudFormation stack that provisions the VPC endpoint
        prerequisite. Waits for CREATE_COMPLETE and returns the VpcEndpointId
        output.

        :return: The VPC endpoint ID.
        """
        print("\n" + "=" * 68)
        print("Step 1: Deploy the CloudFormation prerequisite stack")
        print("=" * 68)
        print(
            f"\nDeploying stack '{self.STACK_NAME}' to create a VPC, subnet, "
            "security group, and syslog VPC endpoint.\n"
            "This can take several minutes..."
        )

        try:
```

```
            self.cf_client.create_stack(
                StackName=self.STACK_NAME,
                TemplateBody=_load_cfn_template(),
                Capabilities=["CAPABILITY_IAM"],
            )
            # Only this run's own stack should be torn down in cleanup. Mark it
            # as soon as creation is initiated so a stack that fails partway
            # (e.g. ROLLBACK) is still cleaned up.
            self.stack_deployed = True
        except ClientError as error:
            if error.response["Error"]["Code"] == "AlreadyExistsException":
                # The stack pre-existed this run. Reuse it but do NOT mark it
                # for cleanup — deleting a stack we did not create could destroy
                # infrastructure the user set up themselves.
                print(
                    f"Stack '{self.STACK_NAME}' already exists. Reusing it; it "
                    "will not be deleted during cleanup."
                )
            else:
                raise

        waiter = self.cf_client.get_waiter("stack_create_complete")
        try:
            waiter.wait(
                StackName=self.STACK_NAME,
                WaiterConfig={"Delay": 30, "MaxAttempts": 40},
            )
        except WaiterError:
            # The waiter fails both when the stack genuinely failed
            # (CREATE_FAILED, ROLLBACK_COMPLETE, ...) and, harmlessly, when the
            # stack already existed in a completed state before this run.
            # Inspect the actual status so a real failure surfaces clearly
            # instead of as a confusing "missing outputs" error later.
            status = self.cf_client.describe_stacks(StackName=self.STACK_NAME)[
                "Stacks"
            ][0]["StackStatus"]
            if status not in ("CREATE_COMPLETE", "UPDATE_COMPLETE"):
                raise RuntimeError(
                    f"CloudFormation stack '{self.STACK_NAME}' did not reach a "
                    f"successful state (status: {status}). Check the stack events
"
                    "in the CloudFormation console for the failure reason."
                )
```

```
        response = self.cf_client.describe_stacks(StackName=self.STACK_NAME)
        outputs = response["Stacks"][0].get("Outputs", list())
        vpc_endpoint_id = None
        for output in outputs:
            if output["OutputKey"] == "VpcEndpointId":
                vpc_endpoint_id = output["OutputValue"]
                break

        if vpc_endpoint_id is None:
            raise RuntimeError(
                "VpcEndpointId not found in stack outputs. "
                "Check the CloudFormation stack for errors."
            )

        self.vpc_endpoint_id = vpc_endpoint_id
        print(f"\nStack deployed. VPC Endpoint ID: {vpc_endpoint_id}")
        return vpc_endpoint_id

    def create_and_verify_log_group(self) -> None:
        """
        Prompts the user for a log group name, creates it, and verifies it
        with DescribeLogGroups.
        """
        print("\n" + "=" * 68)
        print("Step 2: Create and verify a log group")
        print("=" * 68)

        name_pattern = re.compile(r"^[a-zA-Z0-9_\-/.#]{1,512}$")

        while True:
            log_group_name = q.ask(
                "\nEnter a name for the syslog log group (e.g. /syslog/demo): "
            )
            if name_pattern.match(log_group_name):
                break
            print("Invalid name. Use 1-512 characters: a-z, A-Z, 0-9, " "_,
-, /, ., #")

        self.log_group_name = log_group_name
        self.wrapper.create_log_group(log_group_name)

        log_groups = self.wrapper.describe_log_groups(
            log_group_name_prefix=log_group_name
        )
```

```
        for lg in log_groups:
            if lg.get("logGroupName") == log_group_name:
                creation_ms = lg.get("creationTime", 0)
                creation_dt = datetime.fromtimestamp(
                    creation_ms / 1000, tz=timezone.utc
                )
                self.log_group_arn = lg.get("logGroupArn", "")
                print(f"\nLog group verified:")
                print(f"  Name:    {lg['logGroupName']}")
                print(f"  ARN:     {lg.get('arn', 'N/A')}")
                print(f"  Created: {creation_dt.strftime('%Y-%m-%d %H:%M:%S
UTC')}")
                return

        print(f"\nWARNING: Log group '{log_group_name}' not found in describe
results.")

    def add_resource_policy(self) -> None:
        """
        Adds a resource policy to the log group so the syslog service can
        write to it.
        """
        print("\n" + "=" * 68)
        print("Step 3: Add a resource policy for syslog ingestion")
        print("=" * 68)

        if self.log_group_arn is None or self.vpc_endpoint_id is None:
            print(
                "WARNING: Cannot add resource policy - missing log group ARN or
VPC endpoint."
            )
            return

        self.wrapper.put_resource_policy(
            log_group_arn=self.log_group_arn,
            vpc_endpoint_id=self.vpc_endpoint_id,
        )
        print(
            "\nResource policy created. The "
            "syslog.logs.amazonaws.com service principal can now call "
            "logs:PutLogEvents and logs:CreateLogStream on the log group."
        )

    # ----------------------------------------------------------------------
```

```
    # Create syslog configuration phase
    # ------------------------------------------------------------------
    def create_syslog_configuration(self) -> None:
        """
        Creates a syslog configuration that associates the VPC endpoint with
        the log group.
        """
        print("\n" + "=" * 68)
        print("Step 4: Create the syslog configuration")
        print("=" * 68)

        print(
            "\nAssociating the VPC endpoint with the log group so that syslog "
            "messages arriving at the endpoint are stored in the log group."
        )
        print(
            "CloudWatch Logs parses common syslog formats (RFC 5424, "
            "RFC 3164, Cisco FTD/ASA) and extracts fields such as "
            "facility, severity, hostname, and appName.\n"
        )

        self.wrapper.put_syslog_configuration(
            log_group_identifier=self.log_group_name,
            vpc_endpoint_id=self.vpc_endpoint_id,
        )

        print(
            f"\nSyslog ingestion is now enabled for log group "
            f"'{self.log_group_name}' through VPC endpoint "
            f"'{self.vpc_endpoint_id}'."
        )
        print("\nSupported transport options:")
        print("  - TCP with TLS on port 6514 (recommended)")
        print("  - Plaintext TCP on port 1514")
        print("  - UDP on port 514")

    # ------------------------------------------------------------------
    # List syslog configurations phase
    # ------------------------------------------------------------------
    def _display_configurations(self, configs: list) -> None:
        """Helper to display a list of syslog configurations."""
        if not configs:
            print("  (no configurations found)")
            return
```

```
        for cfg in configs:
            created_ms = cfg.get("createdAt", 0)
            created_dt = datetime.fromtimestamp(created_ms / 1000,
tz=timezone.utc)
            print(f"  Log Group ARN:    {cfg.get('logGroupArn', 'N/A')}")
            print(f"  VPC Endpoint ID:  {cfg.get('vpcEndpointId', 'N/A')}")
            print(f"  Source Type:      {cfg.get('sourceType', 'N/A')}")
            print(f"  Created At:       {created_dt.strftime('%Y-%m-%d %H:%M:%S
UTC')}")
            print()

    def list_all_configurations(self) -> None:
        """Lists all syslog configurations in the account (no filter)."""
        print("\n" + "=" * 68)
        print("Step 5a: List all syslog configurations (no filter)")
        print("=" * 68)

        configs = self.wrapper.list_syslog_configurations()
        print(f"\nAll syslog configurations ({len(configs)} found):\n")
        self._display_configurations(configs)

    def list_configurations_by_log_group(self) -> None:
        """Lists syslog configurations filtered by the scenario's log group."""
        print("\n" + "=" * 68)
        print("Step 5b: List configurations filtered by log group")
        print("=" * 68)

        configs = self.wrapper.list_syslog_configurations(
            log_group_identifier=self.log_group_name,
        )
        print(
            f"\nConfigurations for log group '{self.log_group_name}' "
            f"({len(configs)} found):\n"
        )
        self._display_configurations(configs)

    def list_configurations_by_vpc_endpoint(self) -> None:
        """Lists syslog configurations filtered by VPC endpoint."""
        print("\n" + "=" * 68)
        print("Step 5c: List configurations filtered by VPC endpoint")
        print("=" * 68)

        configs = self.wrapper.list_syslog_configurations(
            vpc_endpoint_id=self.vpc_endpoint_id,
```

```
        )
        print(
            f"\nConfigurations for VPC endpoint '{self.vpc_endpoint_id}' "
            f"({len(configs)} found):\n"
        )
        self._display_configurations(configs)

    # ------------------------------------------------------------------
    # Cleanup phase
    # ------------------------------------------------------------------
    def cleanup(self, prompt: bool = True) -> None:
        """
        Cleans up the resources created during the scenario. Tolerates
        resources that were never created or are already gone, so it is safe to
        call after a partial failure.

        :param prompt: When True (the normal end-of-scenario path), ask the user
            to confirm before deleting. When False (best-effort cleanup after an
            error), skip the prompt and delete whatever was created.
        """
        print("\n" + "=" * 68)
        print("Cleanup")
        print("=" * 68)

        if prompt:
            do_cleanup = q.ask(
                "\nDo you want to delete all resources created during this "
                "scenario? (y/n) ",
                q.is_yesno,
            )
            # q.is_yesno converts the answer to a bool (True when the user
            # answered 'y'); check it directly rather than calling string
methods.
            if not do_cleanup:
                print("Skipping cleanup. Resources remain in your account.")
                return
        else:
            print(
                "\nAn error occurred; attempting best-effort cleanup of any "
                "resources that were created..."
            )

        # Delete syslog configuration
        if self.log_group_name and self.vpc_endpoint_id:
```

```
            print(
                f"\nDeleting syslog configuration for log group "
                f"'{self.log_group_name}'..."
            )
            self.wrapper.delete_syslog_configuration(
                log_group_identifier=self.log_group_name,
                vpc_endpoint_id=self.vpc_endpoint_id,
            )
            print(
                "  After deletion, syslog data is no longer ingested through "
                "the VPC endpoint into this log group."
            )

            # Verify deletion
            configs = self.wrapper.list_syslog_configurations(
                log_group_identifier=self.log_group_name,
            )
            if not configs:
                print("  Verified: configuration has been removed.")
            else:
                print(
                    f"  WARNING: Still found {len(configs)} configuration(s). "
                    "They may take a moment to be fully removed."
                )

        # Delete log group
        if self.log_group_name:
            print(f"\nDeleting log group '{self.log_group_name}'...")
            self.wrapper.delete_log_group(self.log_group_name)
            print(
                "  The log group and all archived log events have been "
                "permanently deleted."
            )

        # Delete CloudFormation stack
        if self.stack_deployed:
            print(f"\nDeleting CloudFormation stack '{self.STACK_NAME}'...")
            try:
                self.cf_client.delete_stack(StackName=self.STACK_NAME)
                waiter = self.cf_client.get_waiter("stack_delete_complete")
                waiter.wait(
                    StackName=self.STACK_NAME,
                    WaiterConfig={"Delay": 30, "MaxAttempts": 40},
                )
```

```
                print("  Stack deleted successfully.")
            except (ClientError, WaiterError) as error:
                logger.warning("Stack deletion issue: %s", error)
                print(
                    f"  WARNING: Stack deletion may still be in progress. "
                    f"Check the CloudFormation console."
                )

    # ------------------------------------------------------------------
    # Run the full scenario
    # ------------------------------------------------------------------
    def run_scenario(self) -> None:
        """Runs all phases of the syslog ingestion scenario."""
        print("\n" + "=" * 68)
        print("CloudWatch Logs Syslog Ingestion Scenario")
        print("=" * 68)
        print(
            "\nThis scenario demonstrates CloudWatch Logs managed syslog "
            "ingestion, which lets you route syslog data from firewalls, "
            "routers, switches, and Linux servers directly into CloudWatch "
            "Logs through a VPC endpoint - without running a separate "
            "collection tier.\n"
        )

        succeeded = False
        try:
            # Setup
            self.deploy_cfn_stack()
            self.create_and_verify_log_group()
            self.add_resource_policy()

            # Create syslog configuration
            self.create_syslog_configuration()

            q.ask("\nPress Enter to continue to listing configurations...")

            # List syslog configurations
            self.list_all_configurations()
            self.list_configurations_by_log_group()
            self.list_configurations_by_vpc_endpoint()
            succeeded = True

            # On success, prompt before cleaning up.
            self.cleanup(prompt=True)
```

```
        finally:
            # On failure, run best-effort cleanup without prompting so that
            # partially-created resources are not left behind.
            if not succeeded:
                self.cleanup(prompt=False)

        print("\n" + "=" * 68)
        print("Scenario complete!")
        print("=" * 68)
        print(
            "\nYou have successfully demonstrated CloudWatch Logs managed "
            "syslog ingestion:\n"
            "  - Deployed a CloudFormation stack with a syslog VPC endpoint.\n"
            "  - Created a log group and authorized the syslog service.\n"
            "  - Associated a syslog configuration with PutSyslogConfiguration.
\n"
            "  - Listed configurations with and without filters.\n"
            "  - Cleaned up all resources.\n"
        )
```

创建一个封装 CloudWatch 日志操作的类。

```
class CloudWatchLogsWrapper:
    """Encapsulates Amazon CloudWatch Logs actions."""

    def __init__(self, logs_client: boto3.client) -> None:
        """
        Initializes the CloudWatchLogsWrapper with a CloudWatch Logs client.

        :param logs_client: A Boto3 CloudWatch Logs client.
        """
        self.logs_client = logs_client

    @classmethod
    def from_client(cls) -> "CloudWatchLogsWrapper":
        """
        Creates a CloudWatchLogsWrapper instance with a default CloudWatch Logs
 client.

        :return: An instance of CloudWatchLogsWrapper.
```

```
        """
        logs_client = boto3.client("logs")
        return cls(logs_client)
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的以下主题。
  - CreateLogGroup
  - DeleteLogGroup
  - DeleteSyslogConfiguration
  - DescribeLogGroups
  - ListSyslogConfigurations
  - PutResourcePolicy
  - PutSyslogConfiguration

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 L CloudWatch ogs 与 AWS SDK。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 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。

.NET

适用于 .NET 的 SDK (v4)

> **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。

这是演示大型查询场景的主要工作流程。

```
using System.Diagnostics;
using System.Text.RegularExpressions;
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
```

```
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
using CloudWatchLogsActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace CloudWatchLogsScenario;

public class LargeQueryWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    This .NET code example performs the following tasks for the CloudWatch Logs
 Large Query workflow:

    1. Prepare the Application:
       - Prompt the user to deploy CloudFormation stack and generate sample logs.
       - Deploy the CloudFormation template for resource creation.
       - Generate 50,000 sample log entries using CloudWatch Logs API.
       - Wait 5 minutes for logs to be fully ingested.

    2. Execute Large Query:
       - Perform recursive queries to retrieve all logs using binary search.
       - Display progress for each query executed.
       - Show total execution time and logs found.

    3. Clean up:
       - Prompt the user to delete the CloudFormation stack and all resources.
       - Destroy the CloudFormation stack and wait until removed.
    */

    public static ILogger<LargeQueryWorkflow> _logger = null!;
    public static CloudWatchLogsWrapper _wrapper = null!;
    public static IAmazonCloudFormation _amazonCloudFormation = null!;

    private static string _logGroupName = "/workflows/cloudwatch-logs/large-
query";
    private static string _logStreamName = "stream1";
    private static long _queryStartDate;
    private static long _queryEndDate;

    public static bool _interactive = true;
```

```
    public static string _stackName = "CloudWatchLargeQueryStack";
    private static string _stackResourcePath = "../../../../../../../scenarios/
features/cloudwatch_logs_large_query/resources/stack.yaml";

    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter("Microsoft", LogLevel.Information))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonCloudWatchLogs>()
                    .AddAWSService<IAmazonCloudFormation>()
                    .AddTransient<CloudWatchLogsWrapper>()
            )
            .Build();

        if (_interactive)
        {
            _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
                .CreateLogger<LargeQueryWorkflow>();

            _wrapper = host.Services.GetRequiredService<CloudWatchLogsWrapper>();
            _amazonCloudFormation =
 host.Services.GetRequiredService<IAmazonCloudFormation>();
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query
 Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
 queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to
 retrieve more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));
```

```
            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await ExecuteLargeQuery();
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario,
initiating cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
    }

    /// <summary>
    /// Runs the scenario workflow. Used for testing.
    /// </summary>
    public static async Task RunScenario()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query
Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to
retrieve more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));
```

```
            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await ExecuteLargeQuery();
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario,
initiating cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
    }

    /// <summary>
    /// Prepares the application by creating the necessary resources.
    /// </summary>
    /// <returns>True if the application was prepared successfully.</returns>
    public static async Task<bool> PrepareApplication()
    {
        Console.WriteLine("Preparing the application...");
        Console.WriteLine();

        try
        {
            var deployStack = !_interactive || GetYesNoResponse(
                "Would you like to deploy the CloudFormation stack and generate
sample logs? (y/n) ");

            if (deployStack)
            {
                if (_interactive)
                {
                    Console.Write(
                        $"Enter a path for the CloudFormation stack
resource .yaml file (or press Enter for default '{_stackResourcePath}'): ");
```

```
                    string? inputPath = Console.ReadLine();
                    if (!string.IsNullOrWhiteSpace(inputPath))
                    {
                        _stackResourcePath = inputPath;
                    }
                }

                _stackName = PromptUserForStackName();

                var deploySuccess = await DeployCloudFormationStack(_stackName);

                if (deploySuccess)
                {
                    Console.WriteLine();
                    Console.WriteLine("Generating 50,000 sample log entries...");
                    var generateSuccess = await GenerateSampleLogs();

                    if (generateSuccess)
                    {
                        Console.WriteLine();
                        Console.WriteLine("Sample logs created. Waiting 5 minutes
for logs to be fully ingested...");
                        await WaitWithCountdown(300);

                        Console.WriteLine("Application preparation complete.");
                        return true;
                    }
                }
            }
            else
            {
                _logGroupName = PromptUserForInput("Enter the log group name ",
_logGroupName);
                _logStreamName = PromptUserForInput("Enter the log stream name ",
_logStreamName);

                var startDateMs = PromptUserForLong("Enter the query start date
(milliseconds since epoch): ");
                var endDateMs = PromptUserForLong("Enter the query end date
(milliseconds since epoch): ");

                _queryStartDate = startDateMs / 1000;
                _queryEndDate = endDateMs / 1000;
```

```
                Console.WriteLine("Application preparation complete.");
                return true;
            }
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while preparing the
application.");
        }

        Console.WriteLine("Application preparation failed.");
        return false;
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath)
            };

            var response = await _amazonCloudFormation.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                bool stackCreated = await
WaitForStackCompletion(response.StackId);

                if (stackCreated)
                {
```

```
                    Console.WriteLine("CloudFormation stack created
successfully.");
                    return true;
                }
                else
                {
                    _logger.LogError($"CloudFormation stack creation failed:
{stackName}");
                    return false;
                }
            }
            else
            {
                _logger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            _logger.LogWarning($"CloudFormation stack '{stackName}' already
exists. Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;
```

```
        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };

            var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    return false;
                }
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for CloudFormation stack creation to
complete.");
        return false;
    }

    /// <summary>
    /// Generates sample logs directly using CloudWatch Logs API.
    /// Creates 50,000 log entries spanning 5 minutes.
    /// </summary>
    /// <returns>True if logs were generated successfully.</returns>
    private static async Task<bool> GenerateSampleLogs()
    {
```

```
        const int totalEntries = 50000;
        const int entriesPerBatch = 10000;
        const int fiveMinutesMs = 5 * 60 * 1000;

        try
        {
            // Calculate timestamps
            var startTimeMs = DateTimeOffset.UtcNow.ToUnixTimeMilliseconds();
            var timestampIncrement = fiveMinutesMs / totalEntries;

            Console.WriteLine($"Generating {totalEntries} log entries...");

            var entryCount = 0;
            var currentTimestamp = startTimeMs;
            var numBatches = totalEntries / entriesPerBatch;

            // Generate and upload logs in batches
            for (int batchNum = 0; batchNum < numBatches; batchNum++)
            {
                var logEvents = new List<InputLogEvent>();

                for (int i = 0; i < entriesPerBatch; i++)
                {
                    logEvents.Add(new InputLogEvent
                    {
                        Timestamp =
 DateTimeOffset.FromUnixTimeMilliseconds(currentTimestamp).UtcDateTime,
                        Message = $"Entry {entryCount}"
                    });

                    entryCount++;
                    currentTimestamp += timestampIncrement;
                }

                // Upload batch
                var success = await _wrapper.PutLogEventsAsync(_logGroupName,
 _logStreamName, logEvents);
                if (!success)
                {
                    _logger.LogError($"Failed to upload batch {batchNum + 1}/
{numBatches}");
                    return false;
                }
```

```
                Console.WriteLine($"Uploaded batch {batchNum + 1}/{numBatches}");
            }

            // Set query date range (convert milliseconds to seconds for query
 API)
            _queryStartDate = startTimeMs / 1000;
            _queryEndDate = (currentTimestamp - timestampIncrement) / 1000;

            Console.WriteLine($"Query start date:
 {DateTimeOffset.FromUnixTimeSeconds(_queryStartDate):yyyy-MM-
ddTHH:mm:ss.fffZ}");
            Console.WriteLine($"Query end date:
 {DateTimeOffset.FromUnixTimeSeconds(_queryEndDate):yyyy-MM-ddTHH:mm:ss.fffZ}");
            Console.WriteLine($"Successfully uploaded {totalEntries} log
 entries");

            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while generating sample
 logs.");
            return false;
        }
    }

    /// <summary>
    /// Executes the large query workflow.
    /// </summary>
    public static async Task ExecuteLargeQuery()
    {
        Console.WriteLine("Starting recursive query to retrieve all logs...");
        Console.WriteLine();

        var queryLimit = PromptUserForInteger("Enter the query limit (max 10000)
", 10000);
        if (queryLimit > 10000) queryLimit = 10000;

        var queryString = "fields @timestamp, @message | sort @timestamp asc";

        var stopwatch = Stopwatch.StartNew();
        var allResults = await PerformLargeQuery(_logGroupName, queryString,
 _queryStartDate, _queryEndDate, queryLimit);
        stopwatch.Stop();
```

```
        Console.WriteLine();
        Console.WriteLine($"Queries finished in
{stopwatch.Elapsed.TotalSeconds:F3} seconds.");
        Console.WriteLine($"Total logs found: {allResults.Count}");

        // Check for duplicates
        Console.WriteLine();
        Console.WriteLine("Checking for duplicate logs...");
        var duplicates = FindDuplicateLogs(allResults);
        if (duplicates.Count > 0)
        {
            Console.WriteLine($"WARNING: Found {duplicates.Count} duplicate log
entries!");
            Console.WriteLine("Duplicate entries (showing first 10):");
            foreach (var dup in duplicates.Take(10))
            {
                Console.WriteLine($"  [{dup.Timestamp}] {dup.Message} (appears
{dup.Count} times)");
            }

            var uniqueCount = allResults.Count - duplicates.Sum(d => d.Count -
1);
            Console.WriteLine($"Unique logs: {uniqueCount}");
        }
        else
        {
            Console.WriteLine("No duplicates found. All logs are unique.");
        }
        Console.WriteLine();

        var viewSample = !_interactive || GetYesNoResponse("Would you like to see
a sample of the logs? (y/n) ");
        if (viewSample)
        {
            Console.WriteLine();
            Console.WriteLine($"Sample logs (first 10 of {allResults.Count}):");
            for (int i = 0; i < Math.Min(10, allResults.Count); i++)
            {
                var timestamp = allResults[i].Find(f => f.Field ==
"@timestamp")?.Value ?? "N/A";
                var message = allResults[i].Find(f => f.Field ==
"@message")?.Value ?? "N/A";
                Console.WriteLine($"[{timestamp}] {message}");
```

```
            }
        }
    }

    /// <summary>
    /// Performs a large query using recursive binary search.
    /// </summary>
    private static async Task<List<List<ResultField>>> PerformLargeQuery(
        string logGroupName,
        string queryString,
        long startTime,
        long endTime,
        int limit)
    {
        var queryId = await _wrapper.StartQueryAsync(logGroupName, queryString,
 startTime, endTime, limit);
        if (queryId == null)
        {
            return new List<List<ResultField>>();
        }

        var results = await PollQueryResults(queryId);
        if (results == null || results.Count == 0)
        {
            return new List<List<ResultField>>();
        }

        var startDate =
 DateTimeOffset.FromUnixTimeSeconds(startTime).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var endDate = DateTimeOffset.FromUnixTimeSeconds(endTime).ToString("yyyy-
MM-ddTHH:mm:ss.fffZ");
        Console.WriteLine($"Query date range: {startDate} ({startTime}s) to
 {endDate} ({endTime}s). Found {results.Count} logs.");

        if (results.Count < limit)
        {
            Console.WriteLine($"  -> Returning {results.Count} logs (less than
 limit of {limit})");
            return results;
        }

        Console.WriteLine($"  -> Hit limit of {limit}. Need to split and
 recurse.");
```

```
        // Get the timestamp of the last log (sorted to find the actual last one)
        var lastLogTimestamp = GetLastLogTimestamp(results);
        if (lastLogTimestamp == null)
        {
            Console.WriteLine($"  -> No timestamp found in results. Returning
 {results.Count} logs.");
            return results;
        }

        Console.WriteLine($"  -> Last log timestamp: {lastLogTimestamp}");

        // Parse the timestamp and add 1 millisecond to avoid querying the same
 log again
        var lastLogDate = DateTimeOffset.Parse(lastLogTimestamp + " +0000");
        Console.WriteLine($"  -> Last log as DateTimeOffset: {lastLogDate:yyyy-
MM-ddTHH:mm:ss.fffZ} ({lastLogDate.ToUnixTimeSeconds()}s)");

        var offsetLastLogDate = lastLogDate.AddMilliseconds(1);
        Console.WriteLine($"  -> Offset timestamp (last
 + 1ms): {offsetLastLogDate:yyyy-MM-ddTHH:mm:ss.fffZ}
 ({offsetLastLogDate.ToUnixTimeSeconds()}s)");

        // Convert to seconds, but round UP to the next second to avoid
 overlapping with logs in the same second
        // This ensures we don't re-query logs that share the same second as the
 last log
        var offsetLastLogTime = offsetLastLogDate.ToUnixTimeSeconds();
        if (offsetLastLogDate.Millisecond > 0)
        {
            offsetLastLogTime++; // Move to the next full second
            Console.WriteLine($"  -> Adjusted to next full second:
 {offsetLastLogTime}s
 ({DateTimeOffset.FromUnixTimeSeconds(offsetLastLogTime):yyyy-MM-
ddTHH:mm:ss.fffZ})");
        }

        Console.WriteLine($"  -> Comparing:
 offsetLastLogTime={offsetLastLogTime}s vs endTime={endTime}s");
        Console.WriteLine($"  -> End time as date:
 {DateTimeOffset.FromUnixTimeSeconds(endTime):yyyy-MM-ddTHH:mm:ss.fffZ}");

        // Check if there's any time range left to query
        if (offsetLastLogTime >= endTime)
```

```
        {
            Console.WriteLine($"  -> No time range left to query. Offset time
 ({offsetLastLogTime}s) >= end time ({endTime}s)");
            return results;
        }

        // Split the remaining date range in half
        var (range1Start, range1End, range2Start, range2End) =
 SplitDateRange(offsetLastLogTime, endTime);

        var range1StartDate =
 DateTimeOffset.FromUnixTimeSeconds(range1Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range1EndDate =
 DateTimeOffset.FromUnixTimeSeconds(range1End).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range2StartDate =
 DateTimeOffset.FromUnixTimeSeconds(range2Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range2EndDate =
 DateTimeOffset.FromUnixTimeSeconds(range2End).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");

        Console.WriteLine($"  -> Splitting remaining range:");
        Console.WriteLine($"     Range 1: {range1StartDate} ({range1Start}s) to
 {range1EndDate} ({range1End}s)");
        Console.WriteLine($"     Range 2: {range2StartDate} ({range2Start}s) to
 {range2EndDate} ({range2End}s)");

        // Query both halves recursively
        Console.WriteLine($"  -> Querying range 1...");
        var results1 = await PerformLargeQuery(logGroupName, queryString,
 range1Start, range1End, limit);
        Console.WriteLine($"  -> Range 1 returned {results1.Count} logs");

        Console.WriteLine($"  -> Querying range 2...");
        var results2 = await PerformLargeQuery(logGroupName, queryString,
 range2Start, range2End, limit);
        Console.WriteLine($"  -> Range 2 returned {results2.Count} logs");

        // Combine all results
        var allResults = new List<List<ResultField>>(results);
        allResults.AddRange(results1);
        allResults.AddRange(results2);
```

```
        Console.WriteLine($"  -> Combined total: {allResults.Count} logs
({results.Count} + {results1.Count} + {results2.Count})");

        return allResults;
    }

    /// <summary>
    /// Gets the timestamp string of the most recent log from a list of logs.
    /// Sorts timestamps to find the actual last one.
    /// </summary>
    private static string? GetLastLogTimestamp(List<List<ResultField>> logs)
    {
        var timestamps = logs
            .Select(log => log.Find(f => f.Field == "@timestamp")?.Value)
            .Where(t => !string.IsNullOrEmpty(t))
            .OrderBy(t => t)
            .ToList();

        if (timestamps.Count == 0)
        {
            return null;
        }

        return timestamps[timestamps.Count - 1];
    }

    /// <summary>
    /// Splits a date range in half.
    /// Range 2 starts at midpoint + 1 second to avoid overlap.
    /// </summary>
    private static (long range1Start, long range1End, long range2Start, long
range2End) SplitDateRange(long startTime, long endTime)
    {
        var midpoint = startTime + (endTime - startTime) / 2;
        // Range 2 starts at midpoint + 1 to avoid querying the same second twice
        return (startTime, midpoint, midpoint + 1, endTime);
    }

    /// <summary>
    /// Polls for query results until complete.
    /// </summary>
    private static async Task<List<List<ResultField>>?> PollQueryResults(string
queryId)
```

```
    {
        int retryCount = 0;
        const int maxRetries = 60;
        const int retryDelay = 1000;

        while (retryCount < maxRetries)
        {
            var response = await _wrapper.GetQueryResultsAsync(queryId);
            if (response == null)
            {
                return null;
            }

            if (response.Status == QueryStatus.Complete)
            {
                return response.Results;
            }

            if (response.Status == QueryStatus.Failed ||
                response.Status == QueryStatus.Cancelled ||
                response.Status == QueryStatus.Timeout ||
                response.Status == QueryStatus.Unknown)
            {
                _logger.LogError($"Query failed with status: {response.Status}");
                return null;
            }

            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for query results.");
        return null;
    }

    /// <summary>
    /// Cleans up the resources created during the scenario.
    /// </summary>
    public static async Task<bool> Cleanup()
    {
        var cleanup = !_interactive || GetYesNoResponse(
            "Do you want to delete the CloudFormation stack and all resources?
(y/n) ");
```

```
        if (cleanup)
        {
            try
            {
                var stackDeleteSuccess = await
DeleteCloudFormationStack(_stackName, false);
                return stackDeleteSuccess;
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "An error occurred while cleaning up the
resources.");
                return false;
            }
        }

        Console.WriteLine($"Resources will remain. Stack name: {_stackName}, Log
group: {_logGroupName}");
        _logger.LogInformation("CloudWatch Logs Large Query scenario is
complete.");
        return true;
    }

    /// <summary>
    /// Deletes the CloudFormation stack and waits for confirmation.
    /// </summary>
    private static async Task<bool> DeleteCloudFormationStack(string stackName,
bool forceDelete)
    {
        var request = new DeleteStackRequest
        {
            StackName = stackName,
        };

        if (forceDelete)
        {
            request.DeletionMode = DeletionMode.FORCE_DELETE_STACK;
        }

        await _amazonCloudFormation.DeleteStackAsync(request);
        Console.WriteLine($"CloudFormation stack '{stackName}' is being deleted.
This may take a few minutes.");

        bool stackDeleted = await WaitForStackDeletion(stackName, forceDelete);
```

```
        if (stackDeleted)
        {
            Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
            return true;
        }
        else
        {
            _logger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
            return false;
        }
    }

    /// <summary>
    /// Waits for the stack to be deleted.
    /// </summary>
    private static async Task<bool> WaitForStackDeletion(string stackName, bool
forceDelete)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }
```

```
                if (!forceDelete && describeStacksResponse.Stacks[0].StackStatus
== StackStatus.DELETE_FAILED)
                {
                    return await DeleteCloudFormationStack(stackName, true);
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to
be deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError($"Timed out waiting for CloudFormation stack
'{stackName}' to be deleted.");
        return false;
    }

    /// <summary>
    /// Waits with a countdown display.
    /// </summary>
    private static async Task WaitWithCountdown(int seconds)
    {
        for (int i = seconds; i > 0; i--)
        {
            Console.Write($"\rWaiting: {i} seconds remaining...  ");
            await Task.Delay(1000);
        }
        Console.WriteLine("\rWait complete.                      ");
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
```

```
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompts the user for a stack name.
    /// </summary>
    private static string PromptUserForStackName()
    {
        if (_interactive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press
Enter for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(input))
            {
                var regex = "[a-zA-Z][-a-zA-Z0-9]*";
                if (!Regex.IsMatch(input, regex))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
{_stackName}");
                    return _stackName;
                }
                return input;
            }
        }
        return _stackName;
    }

    /// <summary>
    /// Prompts the user for input with a default value.
    /// </summary>
    private static string PromptUserForInput(string prompt, string defaultValue)
    {
        if (_interactive)
        {
            Console.Write($"{prompt}(press Enter for default '{defaultValue}'):
");
            string? input = Console.ReadLine();
            return string.IsNullOrWhiteSpace(input) ? defaultValue : input;
        }
        return defaultValue;
    }
```

```
    /// <summary>
    /// Prompts the user for an integer value.
    /// </summary>
    private static int PromptUserForInteger(string prompt, int defaultValue)
    {
        if (_interactive)
        {
            Console.Write($"{prompt}(press Enter for default '{defaultValue}'):
");
            string? input = Console.ReadLine();
            if (string.IsNullOrWhiteSpace(input) || !int.TryParse(input, out var
result))
            {
                return defaultValue;
            }
            return result;
        }
        return defaultValue;
    }

    /// <summary>
    /// Prompts the user for a long value.
    /// </summary>
    private static long PromptUserForLong(string prompt)
    {
        if (_interactive)
        {
            Console.Write(prompt);
            string? input = Console.ReadLine();
            if (long.TryParse(input, out var result))
            {
                return result;
            }
        }
        return 0;
    }

    /// <summary>
    /// Finds duplicate log entries based on timestamp and message.
    /// </summary>
    private static List<(string Timestamp, string Message, int Count)>
FindDuplicateLogs(List<List<ResultField>> logs)
    {
```

```
        var logSignatures = new Dictionary<string, int>();

        foreach (var log in logs)
        {
            var timestamp = log.Find(f => f.Field == "@timestamp")?.Value ?? "";
            var message = log.Find(f => f.Field == "@message")?.Value ?? "";
            var signature = $"{timestamp}|{message}";

            if (logSignatures.ContainsKey(signature))
            {
                logSignatures[signature]++;
            }
            else
            {
                logSignatures[signature] = 1;
            }
        }

        return logSignatures
            .Where(kvp => kvp.Value > 1)
            .Select(kvp =>
            {
                var parts = kvp.Key.Split('|');
                return (Timestamp: parts[0], Message: parts[1], Count:
 kvp.Value);
            })
            .OrderByDescending(x => x.Count)
            .ToList();
    }
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API Reference》中的以下主题。
  - GetQueryResults
  - StartQuery

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

await cloudWatchQuery.run();

console.log(
  `Queries finished in ${cloudWatchQuery.secondsElapsed} seconds.\nTotal logs
 found: ${cloudWatchQuery.results.length}`,
);
```

该类可在必要时将查询拆分为多个步骤。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  StartQueryCommand,
  GetQueryResultsCommand,
} from "@aws-sdk/client-cloudwatch-logs";
import { splitDateRange } from "@aws-doc-sdk-examples/lib/utils/util-date.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

class DateOutOfBoundsError extends Error {}

export class CloudWatchQuery {
  /**
   * Run a query for all CloudWatch Logs within a certain date range.
   * CloudWatch logs return a max of 10,000 results. This class
   * performs a binary search across all of the logs in the provided
   * date range if a query returns the maximum number of results.
   *
   * @param {import('@aws-sdk/client-cloudwatch-logs').CloudWatchLogsClient}
 client
   * @param {{ logGroupNames: string[], dateRange: [Date, Date], queryConfig:
 { limit: number } }} config
   */
  constructor(client, { logGroupNames, dateRange, queryConfig }) {
    this.client = client;
    /**
     * All log groups are queried.
     */
    this.logGroupNames = logGroupNames;

    /**
     * The inclusive date range that is queried.
     */
    this.dateRange = dateRange;

    /**
     * CloudWatch Logs never returns more than 10,000 logs.
     */
    this.limit = queryConfig?.limit ?? 10000;

    /**
     * @type {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]}
     */
```

```
    this.results = [];
  }

  /**
   * Run the query.
   */
  async run() {
    this.secondsElapsed = 0;
    const start = new Date();
    this.results = await this._largeQuery(this.dateRange);
    const end = new Date();
    this.secondsElapsed = (end - start) / 1000;
    return this.results;
  }

  /**
   * Recursively query for logs.
   * @param {[Date, Date]} dateRange
   * @returns {Promise<import("@aws-sdk/client-cloudwatch-logs").ResultField[]
[]>}
   */
  async _largeQuery(dateRange) {
    const logs = await this._query(dateRange, this.limit);

    console.log(
      `Query date range: ${dateRange
        .map((d) => d.toISOString())
        .join(" to ")}. Found ${logs.length} logs.`,
    );

    if (logs.length < this.limit) {
      return logs;
    }

    const lastLogDate = this._getLastLogDate(logs);
    const offsetLastLogDate = new Date(lastLogDate);
    offsetLastLogDate.setMilliseconds(lastLogDate.getMilliseconds() + 1);
    const subDateRange = [offsetLastLogDate, dateRange[1]];
    const [r1, r2] = splitDateRange(subDateRange);
    const results = await Promise.all([
      this._largeQuery(r1),
      this._largeQuery(r2),
    ]);
    return [logs, ...results].flat();
```

```
}

/**
 * Find the most recent log in a list of logs.
 * @param {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]} logs
 */
_getLastLogDate(logs) {
  const timestamps = logs
    .map(
      (log) =>
        log.find((fieldMeta) => fieldMeta.field === "@timestamp")?.value,
    )
    .filter((t) => !!t)
    .map((t) => `${t}Z`)
    .sort();

  if (!timestamps.length) {
    throw new Error("No timestamp found in logs.");
  }

  return new Date(timestamps[timestamps.length - 1]);
}

/**
 * Simple wrapper for the GetQueryResultsCommand.
 * @param {string} queryId
 */
_getQueryResults(queryId) {
  return this.client.send(new GetQueryResultsCommand({ queryId }));
}

/**
 * Starts a query and waits for it to complete.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 */
async _query(dateRange, maxLogs) {
  try {
    const { queryId } = await this._startQuery(dateRange, maxLogs);
    const { results } = await this._waitUntilQueryDone(queryId);
    return results ?? [];
  } catch (err) {
    /**
     * This error is thrown when StartQuery returns an error indicating
```

```
       * that the query's start or end date occur before the log group was
       * created.
       */
      if (err instanceof DateOutOfBoundsError) {
        return [];
      }
      throw err;
    }
  }

  /**
   * Wrapper for the StartQueryCommand. Uses a static query string
   * for consistency.
   * @param {[Date, Date]} dateRange
   * @param {number} maxLogs
   * @returns {Promise<{ queryId: string }>}
   */
  async _startQuery([startDate, endDate], maxLogs = 10000) {
    try {
      return await this.client.send(
        new StartQueryCommand({
          logGroupNames: this.logGroupNames,
          queryString: "fields @timestamp, @message | sort @timestamp asc",
          startTime: startDate.valueOf(),
          endTime: endDate.valueOf(),
          limit: maxLogs,
        }),
      );
    } catch (err) {
      /** @type {string} */
      const message = err.message;
      if (message.startsWith("Query's end date and time")) {
        // This error indicates that the query's start or end date occur
        // before the log group was created.
        throw new DateOutOfBoundsError(message);
      }

      throw err;
    }
  }

  /**
   * Call GetQueryResultsCommand until the query is done.
   * @param {string} queryId
```

```
   */
  _waitUntilQueryDone(queryId) {
    const getResults = async () => {
      const results = await this._getQueryResults(queryId);
      const queryDone = [
        "Complete",
        "Failed",
        "Cancelled",
        "Timeout",
        "Unknown",
      ].includes(results.status);

      return { queryDone, results };
    };

    return retry(
      { intervalInMs: 1000, maxRetries: 60, quiet: true },
      async () => {
        const { queryDone, results } = await getResults();
        if (!queryDone) {
          throw new Error("Query not done.");
        }

        return results;
      },
    );
  }
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 AWS SDK API Reference》中的以下主题。
  - GetQueryResults
  - StartQuery

Python

适用于 Python 的 SDK（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来管理超过 10,000 个结果的 CloudWatch 查询。

```
import logging
import os
import sys

import boto3
from botocore.config import Config

from cloudwatch_query import CloudWatchQuery
from date_utilities import DateUtilities

# Configure logging at the module level.
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(filename)s:%(lineno)d - %(message)s",
)

DEFAULT_QUERY_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"


class CloudWatchLogsQueryRunner:
    def __init__(self):
        """
        Initializes the CloudWatchLogsQueryRunner class by setting up date
 utilities
        and creating a CloudWatch Logs client with retry configuration.
        """
        self.date_utilities = DateUtilities()
        self.cloudwatch_logs_client = self.create_cloudwatch_logs_client()

    def create_cloudwatch_logs_client(self):
        """
```

```
        Creates and returns a CloudWatch Logs client with a specified retry
configuration.

        :return: A CloudWatch Logs client instance.
        :rtype: boto3.client
        """
        try:
            return boto3.client("logs", config=Config(retries={"max_attempts":
10}))
        except Exception as e:
            logging.error(f"Failed to create CloudWatch Logs client: {e}")
            sys.exit(1)

    def fetch_environment_variables(self):
        """
        Fetches and validates required environment variables for query start and
end dates.
        Fetches the environment variable for log group, returning the default
value if it
        does not exist.

        :return: Tuple of query start date and end date as integers and the log
group.
        :rtype: tuple
        :raises SystemExit: If required environment variables are missing or
invalid.
        """
        try:
            query_start_date = int(os.environ["QUERY_START_DATE"])
            query_end_date = int(os.environ["QUERY_END_DATE"])
        except KeyError:
            logging.error(
                "Both QUERY_START_DATE and QUERY_END_DATE environment variables
are required."
            )
            sys.exit(1)
        except ValueError as e:
            logging.error(f"Error parsing date environment variables: {e}")
            sys.exit(1)

        try:
            log_group = os.environ["QUERY_LOG_GROUP"]
        except KeyError:
```

```
            logging.warning("No QUERY_LOG_GROUP environment variable, using
 default value")
            log_group = DEFAULT_QUERY_LOG_GROUP

        return query_start_date, query_end_date, log_group

    def convert_dates_to_iso8601(self, start_date, end_date):
        """
        Converts UNIX timestamp dates to ISO 8601 format using DateUtilities.

        :param start_date: The start date in UNIX timestamp.
        :type start_date: int
        :param end_date: The end date in UNIX timestamp.
        :type end_date: int
        :return: Start and end dates in ISO 8601 format.
        :rtype: tuple
        """
        start_date_iso8601 =
 self.date_utilities.convert_unix_timestamp_to_iso8601(
            start_date
        )
        end_date_iso8601 = self.date_utilities.convert_unix_timestamp_to_iso8601(
            end_date
        )
        return start_date_iso8601, end_date_iso8601

    def execute_query(
        self,
        start_date_iso8601,
        end_date_iso8601,
        log_group="/workflows/cloudwatch-logs/large-query",
        query="fields @timestamp, @message | sort @timestamp asc"
    ):
        """
        Creates a CloudWatchQuery instance and executes the query with provided
 date range.

        :param start_date_iso8601: The start date in ISO 8601 format.
        :type start_date_iso8601: str
        :param end_date_iso8601: The end date in ISO 8601 format.
        :type end_date_iso8601: str
        :param log_group: Log group to search: "/workflows/cloudwatch-logs/large-
query"
        :type log_group: str
```

```
        :param query: Query string to pass to the CloudWatchQuery instance
        :type query: str
        """
        cloudwatch_query = CloudWatchQuery(
            log_group=log_group,
            query_string=query
        )
        cloudwatch_query.query_logs((start_date_iso8601, end_date_iso8601))
        logging.info("Query executed successfully.")
        logging.info(
            f"Queries completed in {cloudwatch_query.query_duration} seconds.
 Total logs found: {len(cloudwatch_query.query_results)}"
        )


def main():
    """
    Main function to start a recursive CloudWatch logs query.
    Fetches required environment variables, converts dates, and executes the
 query.
    """
    logging.info("Starting a recursive CloudWatch logs query...")
    runner = CloudWatchLogsQueryRunner()
    query_start_date, query_end_date, log_group =
 runner.fetch_environment_variables()
    start_date_iso8601 = DateUtilities.convert_unix_timestamp_to_iso8601(
        query_start_date
    )
    end_date_iso8601 =
 DateUtilities.convert_unix_timestamp_to_iso8601(query_end_date)
    runner.execute_query(start_date_iso8601, end_date_iso8601,
 log_group=log_group)


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

此模块处理超过 10,000 个结果的 CloudWatch 查询。

```
import logging
import time
from datetime import datetime
```

```
import threading
import boto3

from date_utilities import DateUtilities

DEFAULT_QUERY = "fields @timestamp, @message | sort @timestamp asc"
DEFAULT_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"

class DateOutOfBoundsError(Exception):
    """Exception raised when the date range for a query is out of bounds."""

    pass


class CloudWatchQuery:
    """
    A class to query AWS CloudWatch logs within a specified date range.

    :vartype date_range: tuple
    :ivar limit: Maximum number of log entries to return.
    :vartype limit: int
    :log_group str: Name of the log group to query
    :query_string str: query
    """

    def __init__(self, log_group: str = DEFAULT_LOG_GROUP, query_string:
 str=DEFAULT_QUERY) -> None:
        self.lock = threading.Lock()
        self.log_group = log_group
        self.query_string = query_string
        self.query_results = []
        self.query_duration = None
        self.datetime_format = "%Y-%m-%d %H:%M:%S.%f"
        self.date_utilities = DateUtilities()
        self.limit = 10000

    def query_logs(self, date_range):
        """
        Executes a CloudWatch logs query for a specified date range and
 calculates the execution time of the query.

        :return: A batch of logs retrieved from the CloudWatch logs query.
        :rtype: list
        """
```

```
        start_time = datetime.now()

        start_date, end_date = self.date_utilities.normalize_date_range_format(
            date_range, from_format="unix_timestamp", to_format="datetime"
        )

        logging.info(
            f"Original query:"
            f"\n       START:     {start_date}"
            f"\n       END:       {end_date}"
            f"\n       LOG GROUP: {self.log_group}"
        )
        self.recursive_query((start_date, end_date))
        end_time = datetime.now()
        self.query_duration = (end_time - start_time).total_seconds()

    def recursive_query(self, date_range):
        """
        Processes logs within a given date range, fetching batches of logs
recursively if necessary.

        :param date_range: The date range to fetch logs for, specified as a tuple
(start_timestamp, end_timestamp).
        :type date_range: tuple
        :return: None if the recursive fetching is continued or stops when the
final batch of logs is processed.
                 Although it doesn't explicitly return the query results, this
method accumulates all fetched logs
                 in the `self.query_results` attribute.
        :rtype: None
        """
        batch_of_logs = self.perform_query(date_range)
        # Add the batch to the accumulated logs
        with self.lock:
            self.query_results.extend(batch_of_logs)
        if len(batch_of_logs) == self.limit:
            logging.info(f"Fetched {self.limit}, checking for more...")
            most_recent_log = self.find_most_recent_log(batch_of_logs)
            most_recent_log_timestamp = next(
                item["value"]
                for item in most_recent_log
                if item["field"] == "@timestamp"
            )
            new_range = (most_recent_log_timestamp, date_range[1])
```

```
            midpoint = self.date_utilities.find_middle_time(new_range)

            first_half_thread = threading.Thread(
                target=self.recursive_query,
                args=((most_recent_log_timestamp, midpoint),),
            )
            second_half_thread = threading.Thread(
                target=self.recursive_query, args=((midpoint, date_range[1]),)
            )

            first_half_thread.start()
            second_half_thread.start()

            first_half_thread.join()
            second_half_thread.join()

    def find_most_recent_log(self, logs):
        """
        Search a list of log items and return most recent log entry.
        :param logs: A list of logs to analyze.
        :return: log
        :type :return List containing log item details
        """
        most_recent_log = None
        most_recent_date = "1970-01-01 00:00:00.000"

        for log in logs:
            for item in log:
                if item["field"] == "@timestamp":
                    logging.debug(f"Compared: {item['value']} to
{most_recent_date}")
                    if (
                        self.date_utilities.compare_dates(
                            item["value"], most_recent_date
                        )
                        == item["value"]
                    ):
                        logging.debug(f"New most recent: {item['value']}")
                        most_recent_date = item["value"]
                        most_recent_log = log
        logging.info(f"Most recent log date of batch: {most_recent_date}")
        return most_recent_log

    def perform_query(self, date_range):
```

```
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_group,
                    startTime=start_time,
                    endTime=end_time,
                    queryString=self.query_string,
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []
```

```
    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")


    def _wait_for_query_results(self, client, query_id):
        """
        Waits for the query to complete and retrieves the results.

        :param query_id: The ID of the initiated query.
        :type query_id: str
        :return: A list containing the results of the query.
        :rtype: list
        """
        while True:
            time.sleep(1)
```

```
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",
                "Cancelled",
                "Timeout",
                "Unknown",
            ]:
                return results.get("results", [])
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》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，请参阅将 L CloudWatch ogs 与 AWS SDK。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。

使用计划的事件调用 Lambda 函数

以下代码示例演示如何创建由 Amazon EventBridge 计划事件调用的 AWS Lambda 函数。

Java

适用于 Java 的 SDK 2.x

演示如何创建调用函数的 Amazon EventBridge 计划事件。 AWS Lambda 配置 EventBridge 为使用 cron 表达式来调度 Lambda 函数的调用时间。在本示例中，您使用 Lambda Java 运行时 API 创建 Lambda 函数。此示例调用不同的 AWS 服务来执行特定的用例。此示例展示了如何创建一个应用程序，在其一周年纪念日时向员工发送移动短信表示祝贺。

有关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
- DynamoDB
- EventBridge
- Lambda

- Amazon SNS

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

演示如何创建调用函数的 Amazon EventBridge 计划事件。 AWS Lambda 配置 EventBridge 为使用 cron 表达式来调度 Lambda 函数的调用时间。在此示例中，您将使用 Lambda 运行时 API 创建一个 Lambda 函数。 JavaScript 此示例调用不同的 AWS 服务来执行特定的用例。此示例展示了如何创建一个应用程序，在其一周年纪念日时向员工发送移动短信表示祝贺。

有关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。

该示例也可在 适用于 JavaScript 的 AWS SDK v3 开发人员指南中找到。

本示例中使用的服务

- CloudWatch 日志
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

适用于 Python 的 SDK（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
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 L CloudWatch ogs 与 AWS SDK。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。

使用加密 CloudWatch 日志中的查找表 AWS Key Management Service

Logs 中始终对查找表数据进行 CloudWatch 加密。默认情况下， CloudWatch Logs 使用具有 256 位高级加密标准 Galois/Counter 模式 (AES-GCM) 的服务器端加密来加密查询表的静态数据。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。在查找表级别启用加密，方法 AWS KMS 是在创建查找表时或更新查找表时将 KMS 密钥与查找表关联。

> **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找表中的数据。有关更多信息，请参阅使用对称和非对称密钥。

将 KMS 密钥与查找表关联后，存储在查找表中的所有数据都将使用此密钥进行加密。 CloudWatch 每当有人请求这些数据时，日志都会对其进行解密。 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 密钥与查找表的关联，Logs 将使用 CloudWatch CloudWatch 日志默认加密方法对数据进行加密。但是，如果在解除关联之前密钥被禁用或删除，则 CloudWatch Logs 将无法读取使用该密钥加密的数据。

有关 L CloudWatch ogs AWS KMS 如何使用加密日志数据的一般信息，请参阅使用加密日志中的 CloudWatch 日志数据 AWS Key Management Service。

CloudWatch 日志如何 AWS KMS 用于查找表

CloudWatch 日志使用 AWS KMS 信封加密来保护查找表数据。当您将 KMS 密钥与查找表关联时，L CloudWatch ogs 会向发送`GenerateDataKey`请求 AWS KMS。 AWS KMS 生成唯一的数据加密密钥 (DEK) 并返回 DEK 的纯文本副本和加密副本。 CloudWatch 日志使用纯文本 DEK 对查找表数据进行加密，然后将加密的 DEK 与加密数据一起存储。纯文本 DEK 不存储，使用后会从内存中丢弃。

当 CloudWatch Logs 需要读取查找表数据时，它会 AWS KMS 使用加密的 DEK 向发送`Decrypt`请求。 AWS KMS 解密 DEK 并将纯文本 DEK 返回到 Logs，然后 CloudWatch 日志使用它来解密查找表数据。

CloudWatch 在向日志发出请求时，日志使用以下加密上下文 AWS KMS：

```
{
    "aws:logs:arn": "arn:aws:logs:region:account-id:lookup-table:lookup-table-name"
}
```

您可以在 IAM 策略和密 AWS KMS 钥策略中使用此加密上下文来控制对 KMS 密钥的访问。有关更多信息，请参阅 AWS KMS 密钥和加密上下文。

所需的权限

要对查找表使用 AWS KMS 加密，IAM 委托人必须对 KMS 密钥 AWS KMS 具有以下权限：

- `kms:Decrypt`
- `kms:GenerateDataKey`

调用`GetLookupTable`使用 KMS 密钥加密的查找表时需要该`kms:Decrypt`权限，这样 L CloudWatch ogs 才能代表您解密数据。当使用对加密的查找表使用`lookup`命令的查询进行调`StartQuery`用时，还需要对密钥（用于加密查找表的 KMS 密钥）拥有`kms:Decrypt`权限。调`UpdateLookupTable`用`CreateLookupTable`或使用 KMS 密钥时需要该`kms:GenerateDataKey`权限，这样 L CloudWatch ogs 才能生成数据加密密钥来加密查找表数据。

此外，CloudWatch 日志服务必须有权使用 KMS 密钥。您可以通过向 KMS 密钥策略添加策略声明来授予这些权限，如下一节所述。

步骤 1：创建 AWS KMS 密钥

要创建对称 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"
        ]
    }
}
```

记录密钥 ARN。你需要在以下步骤中使用它。

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。您必须向 CloudWatch 日志服务主体授予使用密钥的权限，还必须向调用角色授予使用密钥的权限。

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`文件并添加以下语句以授予 CloudWatch 日志服务主体使用密钥的权限。此示例使用与加密上下文匹配的`Condition`部分将密钥限制在特定的查找表中。

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "logs.region.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
```

```
            "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:region:account-
id:lookup-table:lookup-table-name",
            "aws:SourceAccount": "account-id",
            "aws:SourceArn": "arn:aws:logs:region:account-id:lookup-table:lookup-table-
name"
        }
    }
}
```

接下来，为将调用 CloudWatch 日志的角色添加权限`CreateLookupTable`或`UpdateLookupTable` APIs。 CloudWatch 记录`kms:ViaService`用户 AWS KMS 代表客户拨打电话。有关更多信息，请参阅 kms: ViaService。

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::account-id:role/role-name"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:ViaService": [
                "logs.region.amazonaws.com"
            ]
        }
    }
}
```

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 密钥与查找表关联

在使用 `CreateLookupTable` API 创建 KMS 密钥时，您可以将其与查找表关联，也可以使用 `UpdateLookupTable` API 更新现有的查找表。两者 APIs 都是 AWSLogsConfigService.

在创建 KMS 密钥时将其与查找表关联

使用 CreateLookupTable API 并使用您的 KMS 密钥的 ARN 指定kmsKeyArn参数：

```
aws logs create-lookup-table \
    --lookup-table-name my-lookup-table \
    --kms-key-arn "arn:aws:kms:region:account-id:key/key-id"
```

将 KMS 密钥与现有查找表关联

使用 UpdateLookupTable API 并使用您的 KMS 密钥的 ARN 指定kmsKeyArn参数：

```
aws logs update-lookup-table \
    --lookup-table-name my-lookup-table \
    --kms-key-arn "arn:aws:kms:region:account-id:key/key-id"
```

注意事项

- 将 KMS 密钥与查找表关联或取消关联后，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 日志中的加密查找表数据。
- 要执行本主题中的步骤，您必须具有以下权限：kms:CreateKey、kms:GetKeyPolicykms:PutKeyPolicy、以及相应的 CloudWatch Logs 权限才能调用CreateLookupTable或UpdateLookupTable。

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
- 亚马逊 CloudWatch 日志的身份和访问管理
- 亚马逊 CloudWatch 日志的合规性验证
- Amazon CloudWatch 日志中的弹性
- Amazon CloudWatch 日志中的基础设施安全
- 将 CloudWatch 日志与接口 VPC 终端节点配合使用

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结构上的内容的控制。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解答AWS条款。 有关欧洲数据保护的信息，请参阅通用数据保护条例（GDPR）中心。

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。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）。
- 用于 SSL/TLS 与 AWS 资源通信。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。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅《AWS CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 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 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》https://aws.amazon.com/compliance/fips/。

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 URL，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，并使用具有 256 位高级加密标准 Galois/Counter 模式 (AES-GCM) 的服务器端加密来加密静态日志数据。

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 日志对传输中的数据使用端到端加密。 CloudWatch 日志服务管理服务器端的加密密钥。

亚马逊 CloudWatch 日志的身份和访问管理

访问亚马逊 CloudWatch 日志需要 AWS 可用于对您的请求进行身份验证的证书。这些凭据必须具有访问 AWS 资源的权限，例如检索有关您的云资源的 CloudWatch 日志数据。以下部分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您的资源来帮助保护您的资源：

- 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：

- 用户和群组位于 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。
- 通过身份提供者在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中针对第三方身份提供者创建角色（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据来对请求进行身份验证，但除非您有权限，否则无法创建或访问 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。

以下部分介绍如何管理 CloudWatch 日志的权限。我们建议您先阅读概述。

- 管理 CloudWatch 日志资源访问权限的概述
- 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 CloudWatch 日志资源访问权限的概述

要提供访问权限，请为您的用户、组或角色添加权限：

- 用户和群组位于 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。
- 通过身份提供者在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中针对第三方身份提供者创建角色（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（用于主要资源的其他资源）。

这些资源和子资源具有与其关联的唯一 Amazon 资源名称（ARN），如下表所示。

资源类型	ARN 格式
日志组	arn: aws:: logs:: log-group: *region account-id log_group_name*

资源类型	ARN 格式
日志流	arn: aws: logs:: log-group:: log-stream: *region account-id log_group_name log-stream-name*
目标位置	arn: aws: logs::: 目的地:*regionaccount-id 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，请参阅中的亚马逊资源名称 (ARN) 。Amazon Web Services 一般参考有关涵盖 CloudWatch 日志的策略示例，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，无论资源是谁创建的。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据创建日志组，则您的 AWS 账户是 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，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 日志资源权限的 IAM 角色，则任何可以代入该角色的人都可以创建 CloudWatch 日志资源。该角色所属的 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针对目标的基于资源的策略，这些策略用于启用跨账户订阅。有关更多信息，请参阅 Cross-account 跨区域订阅。

主题

- 日志组权限和 Contributor Insights
- Resource-based 政策

日志组权限和 Contributor Insights

贡献者见解是一项功能 CloudWatch ，使您能够分析日志组中的数据并创建显示贡献者数据的时间序列。您可以查看有关前 N 个贡献者、独特贡献者总数及其使用情况的指标。有关更多信息，请参阅使用投稿人见解分析 High-Cardinality 数据。

贡献者见解操作使用 `cloudwatch:` IAM 命名空间。 CloudWatch 当贡献者见解处理日志组数据时，不会评估 Logs (`logs:`) 权限。有关贡献者见解权限模型、条件密钥和最佳实践的详细信息，请参阅《亚马逊用户指南》中的使用条件密钥限制贡献者见解 CloudWatch 用户对日志组的访问权限。

如果您的日志组包含敏感数据，请在贡献者见解或任何其他功能对其进行处理之前，使用 CloudWatch 日志数据保护策略来屏蔽这些数据。有关更多信息，请参阅 通过屏蔽帮助保护敏感的日志数据。

Resource-based 政策

CloudWatch 日志支持基于资源的目标策略，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 PutDestination API 创建目的地，也可以使用 PutDestinationPolicy API 向目标添加资源策略。在以下示例中，允许账户 ID 为 111122223333 的另一个 AWS 账户将其日志组订阅到目标 `arn:aws:logs:us-east-1:123456789012:destination:testDestination`。

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111122223333"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-
east-1:123456789012:destination:testDestination"
    }
  ]
}
```

指定策略元素：操作、效果和主体

对于每个 CloudWatch 日志资源，该服务定义了一组 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 资源名称（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 策略语法和描述的更多信息，请参阅《IAM 用户指南》中的 AWS IAM 策略参考。

有关显示所有 CloudWatch 日志 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AWS 局条件上下文密钥的列表，请参阅 AWS 服务的操作、资源和条件密钥。

> **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，这些主题解释了可用于管理 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 CloudWatch 日志资源访问权限的概述。

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：

JSON

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

本策略具有一个语句，该语句授予了创建日志组和日志流、将事件上传到日志流和列出有关日志流的详细信息的权限。

Resource值中的通配符 (*) 授予对任何 CloudWatch 日志资源执行列出的操作的权限。

- 要限制特定日志组操作的权限（例如，，DescribeLogStreams）CreateLogGroup，请将通配符替换为日志组 ARN— arn:aws:logs:*us-west-2*:*123456789012*:log-group:*MyLogGroup*
- 要限制特定日志流操作（例如，CreateLogStream、PutLogEvents）的权限，请将通配符替换为日志流 ARN—arn:aws:logs:*us-west-2*:*123456789012*:log-group:*MyLogGroup*:*（匹配所有流）或arn:aws:logs:*us-west-2*:*123456789012*:log-group:*MyLogGroup*:log-stream:*MyStream*（特定流）

有关每个 API 所需的资源类型，请参阅服务授权参考。

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`托管策略附加到用户，如中所述AWS CloudWatch 日志的托管（预定义）策略。

您无需为仅调用 AWS CLI 或 CloudWatch 日志 API 的用户提供最低控制台权限。

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
- 云表：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
- kinesis：DescribeStreams
- kinesis：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。托管策略可针对常见使用案例授予必要权限，因此，您无需自行调查具体需要哪些权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

您可以将以下 AWS 托管策略附加到账户中的用户和角色，这些策略特定于 CloudWatch 日志：

- 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政策授予对 CloudWatch 日志的完全访问权限。该策略包括`cloudwatch:GenerateQuery`和`cloudwatch:GenerateQueryResultsSummary`权限，因此使用此策略的用户可以根据自然语言提示生成 L CloudWatch ogs Insights 查询字符串。要查看策略的完整内容，请参阅CloudWatchLogsFullAccess《AWS 托管策略参考指南》。

CloudWatchLogsReadOnlyAccess

该CloudWatchLogsReadOnlyAccess策略授予对 CloudWatch 日志的只读访问权限。它包括`cloudwatch:GenerateQuery`和`cloudwatch:GenerateQueryResultsSummary`权限，因此具有此策略的用户可以根据自然语言提示生成 L CloudWatch ogs Insights 查询字符串。要查看策略的完整内容，请参阅CloudWatchLogsReadOnlyAccess《AWS 托管策略参考指南》。

CloudWatchOpenSearchDashboardsFullAccess

该CloudWatchOpenSearchDashboardsFullAccess政策授予创建、管理和删除与 Serv OpenSearch ice 集成的权限，以及在这些集成中创建、删除和管理已售日志仪表板的权限。

要查看策略的完整内容，请参阅CloudWatchOpenSearchDashboardsFullAccess《AWS 托管策略参考指南》。

CloudWatchOpenSearchDashboardAccess

该CloudWatchOpenSearchDashboardAccess政策授予查看通过 Amazon OpenSearch Service 分析创建的销售日志仪表板的访问权限。

要查看策略的完整内容，请参阅CloudWatchOpenSearchDashboardAccess《AWS 托管策略参考指南》。

CloudWatchLogsCrossAccountSharingConfiguration

该CloudWatchLogsCrossAccountSharingConfiguration政策授予创建、管理和查看用于在账户之间共享 CloudWatch 日志资源的可观察性访问管理器链接的权限。有关更多信息，请参阅 CloudWatch 跨账户可观测性。

要查看策略的完整内容，请参阅CloudWatchLogsCrossAccountSharingConfiguration《AWS 托管策略参考指南》。

CloudWatchLogsAPIKeyAccess

该CloudWatchLogsAPIKeyAccess策略启用 CloudWatch 日志 API 密钥身份验证和加密日志提取。此策略授予使用持有者令牌进行身份验证以及将日志事件写入CloudWatch 日志的 AWS KMS 权限，以及在加密日志时解密和生成数据密钥的额外权限。

此策略授予以下权限：

- `logs`— 允许委托人通过 API 密钥持有者令牌进行身份验证并将日志事件写入日志流 CloudWatch 。
- `kms`— 允许委托人读取 AWS KMS 密钥元数据、生成用于加密的数据密钥和解密数据。这些权限允许服务使用客户管理的 AWS KMS 密钥加密日志数据，从而支持加密日志。 CloudWatch 访问权限仅限于通过 CloudWatch 日志服务调用的操作。

要查看有关策略（包括 JSON 策略文档的最新版本）的更多信息，请参阅《AWS 托管式策略参考指南》中的 CloudWatchLogsAPIKeyAccess。

CloudWatch 将更新记录到 AWS 托管策略

查看自该服务开始跟踪这些更改以来 CloudWatch 日志 AWS 管理策略更新的详细信息。要自动收到有关此页面更改的警报，请订阅 CloudWatch 日志文档历史记录页面上的 RSS 提要。

更改	描述	日期
CloudWatchLogsAPIKeyAccess – 新策略。	CloudWatch Logs 添加了新的托管策略CloudWatchLogsAPIKeyAccess。 此策略启用 CloudWatch 日志 API 密钥身份验证和加密日志提取，授予使用持有者令牌进行身份验证以及向日志写入日志事件的权限。 CloudWatch	2026 年 2 月 17 日
CloudWatchLogsFullAccess – 对现有策略的更新。	CloudWatch 日志向添加了权限CloudWatchLogsFullAccess。 添加了可观测性管理操作的权限，以允许对遥测管道及 S3 表集成进行只读访问。	2025 年 12 月 2 日
CloudWatchLogsReadOnlyAccess – 对现有策略的更新。	CloudWatch 日志向添加了权限CloudWatchLogsReadOnlyAccess。 添加了可观测性管理操作的权限，以允许对遥测管道及 S3 表集成进行只读访问。	2025 年 12 月 2 日
CloudWatchLogsFullAccess – 对现有策略的更新。		2025 年 5 月 20 日

更改	描述	日期
	CloudWatch 日志向添加了权限CloudWatchLogsFullAccess。 添加了 `cloudwatch:GenerateQueryResultsSummary` 的权限，以允许生成查询结果的自然语言摘要。	
CloudWatchLogsReadOnlyAccess – 对现有策略的更新。	CloudWatch 日志向添加了权限CloudWatchLogsReadOnlyAccess。 添加了 `cloudwatch:GenerateQueryResultsSummary` 的权限，以允许生成查询结果的自然语言摘要。	2025 年 5 月 20 日
CloudWatchLogsFullAccess – 对现有策略的更新。	CloudWatch 日志向添加了权限CloudWatchLogsFullAccess。 添加了 Amazon OpenSearch Service 和 IAM 的权限，以启用某些功能的 CloudWatch 日志与 OpenSearch 服务的集成。	2024 年 12 月 1 日

更改	描述	日期
CloudWatchOpenSearchDashboardsFullAccess— 新的 IAM 政策。	CloudWatch Logs 添加了新的 IAM 策略CloudWatchOpenSearchDashboardsFullAccess。-该策略授予创建、管理和删除与 Serv OpenSearch ice 集成的权限，以及在这些集成中创建、管理和删除已售日志仪表板的权限。	2024 年 12 月 1 日
CloudWatchOpenSearchDashboardAccess— 新的 IAM 政策。	CloudWatch Logs 添加了新的 IAM 策略CloudWatchOpenSearchDashboardAccess。-该策略授予查看由 Amazon OpenSearch Service 其提供支持的供应商日志仪表板的访问权限。	2024 年 12 月 1 日
CloudWatchLogsFullAccess – 对现有策略的更新。	CloudWatch 日志向添加了权限CloudWatchLogsFullAccess。 添加了`cloudwatch:GenerateQuery` 权限，因此具有此策略的用户可以根据自然语言提示生成 L CloudWatch ogs Insights 查询字符串。	2023 年 11 月 27 日

更改	描述	日期
CloudWatchLogsReadOnlyAccess – 对现有策略的更新。	CloudWatch 向... 添加了权限CloudWatchLogsReadOnlyAccess。 添加了`cloudwatch:GenerateQuery` 权限，因此具有此策略的用户可以根据自然语言提示生成 L CloudWatch ogs Insights 查询字符串。	2023 年 11 月 27 日
CloudWatchLogsReadOnlyAccess：对现有策略的更新	CloudWatch 日志向添加了权限CloudWatchLogsReadOnlyAccess。 添加`logs:StartLiveTail` 和权`logs:StopLiveTail` 限是为了使用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
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。 有关更多信息，请参阅 CloudWatch 跨账户可观察性	2022 年 11 月 27 日

更改	描述	日期
CloudWatchLogsReadOnlyAccess：对现有策略的更新	CloudWatch 日志向添加了权限CloudWatchLogsReadOnlyAccess。 添加了`oam:ListSinks` 和`oam:ListAttachedLinks` 权限，这样具有此策略的用户就可以使用控制台查看 CloudWatch 跨账户可观察性中源账户共享的数据。	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/日志流

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。

JSON

```
{
  "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。

JSON

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

示例 3：允许访问一个日志组/日志流

CloudWatch 日志有两种具有不同 ARN 格式的资源类型：

- 日志组：arn:aws:logs:*region*:*account*:log-group:*LogGroupName*
- 日志流：arn:aws:logs:*region*:*account*:log-group:*LogGroupName*:log-stream:*StreamName*

在编写 IAM 策略时，您使用的 ARN 格式必须与 API 授权的资源类型相匹配。有关每个 API 所需的资源类型，请参阅服务授权参考。

示例 3a：允许访问特定日志组上的日志组级操作

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
      "Action": [
        "logs:DeleteLogGroup",
        "logs:PutRetentionPolicy",
        "logs:PutSubscriptionFilter",
        "logs:DescribeLogStreams"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:us-west-2:123456789012:log-group:SampleLogGroupName"
      }
   ]
}
```

这些操作对log-group资源类型进行授权。支持标准 ARN 格式（不支持:*）。

示例 3b：允许访问特定日志组上的日志流级别操作

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

```
      "Effect": "Allow",
      "Resource": "arn:aws:logs:us-west-2:123456789012:log-group:SampleLogGroupName:*"
      }
   ]
}
```

这些操作对log-stream资源类型进行授权。必须使用:*后缀来匹配日志组中的所有日志流，或者使用:log-stream:*StreamName*指定特定的流。

示例 3c：日志组和日志流操作的组合策略

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
        "Action": [
          "logs:DeleteLogGroup",
          "logs:PutRetentionPolicy",
          "logs:DescribeLogStreams",
          "logs:FilterLogEvents"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:logs:us-west-2:123456789012:log-group:SampleLogGroupName"
      },
      {
        "Action": [
          "logs:CreateLogStream",
          "logs:PutLogEvents",
          "logs:GetLogEvents"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:logs:us-west-2:123456789012:log-
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。

JSON

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

StopQuery和 StopLiveTail API 操作不与传统意义上的 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，这些查询不被归类为资源。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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
        [ {
            "Effect": "Allow",
            "Action": [
                "logs:StopQuery",
                "logs:StopLiveTail"
```

```
            ],
            "Resource": "*"
            }
        ]
}
```

有关使用 IAM policy 语句的更多信息，请参阅 IAM 用户指南中的使用策略控制访问。

CloudWatch 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。该表列出了每个 L CloudWatch ogs API 操作以及您可以授予执行操作权限的相应操作。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全局密钥的完整列表，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 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。

CloudWatch 记录 API 操作	所需权限（API 操作）
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。
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 L CloudWatch ogs Insights 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。

CloudWatch 记录 API 操作	所需权限（API 操作）
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。
DescribeQueryDefinitions	`logs:DescribeQueryDefinitions` 需要在 L CloudWatch ogs Insights 中查看已保存的查询定义列表。

CloudWatch 记录 API 操作	所需权限（API 操作）
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` 需要查看 CloudWatch 日志资源策略列表。
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。

CloudWatch 记录 API 操作	所需权限（API 操作）
GetLogObject	`logs:GetLogRecord` 需要获取通过 PutOpenTelemetryLogs API 摄取的大部分日志事件的内容。
GetQueryResults	`logs:GetQueryResults` 检索 CloudWatch 日志见解查询结果所必需的。
ListEntitiesForLogGroup （CloudWatch 仅限控制台权限）	`logs:ListEntitiesForLogGroup` 查找与日志组关联的实体所需的权限。需要在 CloudWatch 控制台中浏览相关日志。
ListLogGroupsForEntity （CloudWatch 仅限主机权限）	`logs:ListLogGroupsForEntity` 查找与实体关联的日志组所需的权限。需要在 CloudWatch 控制台中浏览相关日志。
ListTagsLogGroup	`logs:ListTagsLogGroup` 需要列出与日志组关联的标签。
ListLogGroups	`logs:DescribeLogGroups` 需要查看与账户关联的所有日志组。
ProcessWithPipeline（仅限许可）	`logs:ProcessWithPipeline` 需要在存储之前通过管道转换器处理和转换日志事件。必须对在管道配置中作为 CloudWatch 日志源角色传递的 IAM 角色授予此权限。资源是日志组 ARN。

CloudWatch 记录 API 操作	所需权限（API 操作）
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 L CloudWatch ogs Insights 中保存查询，包括已保存的带参数的查询。
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 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` 需要启动 CloudWatch Logs Insights 查询。要使用参数运行已保存的查询，您还需要`logs:DescribeQueryDefinitions` 。
StopQuery	`logs:StopQuery` 需要停止正在进行的 CloudWatch 日志见解查询。
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

亚马逊 CloudWatch 日志使用 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，直接关联到 CloudWatch 日志。 Service-linked 角色由 L CloudWatch ogs 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设置 CloudWatch 日志的效率，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 L CloudWatch ogs 可以代入这些角色。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。在 “Service-Linked 角色” 列中查找 “是” 的服务。请选择是与查看该服务的服务关联角色文档的链接。

Service-linked CloudWatch 日志的角色权限

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到 Firehose。有关更多信息，请参阅 启用从的日志记录 AWS 务。

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 日志在指定资源上完成以下操作。AWSServiceRoleForLogDeliveryPolicy 有关完整的 JSON 策略文档，请参阅AWSServiceRoleForLogDeliveryPolicy《AWS 托管策略参考指南》。

- `firehose:ListTagsForDeliveryStream`在所有带有标签且`LogDeliveryEnabled`密钥值为的 Firehose 交付流上执行操作：`firehose:PutRecord``firehose:PutRecordBatch`、和。`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 传送流。
- 操作：`kms:GenerateDataKey``kms:Decrypt`对所有 AWS KMS 密钥执行操作，但仅限于通过 Firehose 发出请求（由`kms:ViaService`条件键设置为强制执行`firehose.*.amazonaws.com`）。这些权限允许CloudWatch 日志将日志传送到使用服务器端加密和客户管理密钥的 Firehose 交付流 () SSE-CMK。客户的 AWS KMS 密钥策略还必须独立授予对服务相关角色的访问权限。

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Service-Linked 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。当您将日志设置为直接发送到 AWS 管理控制台、或 AWS API 中的 Firehose 直播时 AWS CLI，L CloudWatch ogs 会为您创建服务相关角色。

如果您删除该服务关联角色，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 直播时，L CloudWatch ogs 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 创建后 AWSServiceRoleForLogDelivery，日志不允许您对其进行编辑或任何其他服务相关角色。由于多个实体可能引用该角色，因此无法更改角色的名称。不过，您可以使用 IAM 编辑角色的说明。有关更多信息，请参阅 IAM 用户指南中的编辑 Service-Linked 角色。

删除日志的服务相关角色 CloudWatch

如果您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。这样您就没有未被主动监控或维护的未使用实体。但是，您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。

删除使用的 CloudWatch 日志资源 AWSServiceRoleForLogDelivery 服务相关角色

- 停止将日志直接发送到 Firehose 流。

使用 IAM 手动删除服务关联角色

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 Service-Linked 角色

CloudWatch 日志服务相关角色支持的区域

CloudWatch 日志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 日志区域和终端节点。

CloudWatch 将更新记录到 AWS 服务相关角色

查看自该 AWS 服务开始跟踪这些更改以来 CloudWatch 日志的服务关联角色更新的详细信息。要自动收到有关此页面更改的警报，请订阅 CloudWatch 日志文档历史记录页面上的 RSS 提要。

更改	描述	日期
AWSServiceRoleForLogDelivery 服务相关角色策略 - 更新现有策略	CloudWatch 添加的日志`kms:GenerateDataKey` 和与AWSServiceRoleForLogDelivery服务相关角色关联的 IAM 策略的`kms:Decrypt` 权限。这些权限的范围以`kms:ViaService` 条件密钥为限，仅允许通过 Firehose 使用。此更改使 CloudWatch 日志能够将日志传送到使用服务器端加密和客户管理密钥的 Firehose 交付流 () SSE-CMK。	2026 年 5 月 15 日
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"` 条件键已更改为 `aws:ResourceTag/LogDeliveryEnabled": "true"` 。	2021 年 7 月 15 日
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

亚马逊 CloudWatch 日志的合规性验证

我们的新 AWS 注册体验不是为受监管的工作负载设计的。如果您正在使用我们的全新 AWS 注册体验，但想 AWS 用于受监管的工作负载，则可以注册 AWS （高级）或激活 AWS 环境的高级功能。

要了解是否属于特定合规计划的范围，请参阅 “按合规计划 AWS 服务 中的 “按范围划分的合规计划”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划。

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 AWS 服务 时的合规责任取决于数据的敏感性、贵公司的合规目标以及适用的法律法规。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅AWS 安全文档。

Amazon CloudWatch 日志中的弹性

AWS 全球基础设施是围绕 AWS 区域和可用区构建的。各区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错能力和可扩展性。

有关 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。

将 CloudWatch 日志与接口 VPC 终端节点配合使用

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 Internet 发送日志。 CloudWatch 日志支持所有区域 IPv4 的 VPC 终端节点，并支持所有区域的 IPv6 终端节点。

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，请参阅新增- AWS PrivateLink 适用于 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。可供选择的服务是 com.amazonaws。 ***Region***.logs。要连接 FIPS 终端节点，需要选择的服务是`com.amazonaws.Region.logs-fips`。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。

某些 CloudWatch 日志 APIs（例如 StartLiveTail 和 GetLogObject）托管在不同的终端节点和 VPC 终端节点下：`stream-logs.Region.amazonaws.com`。要为它们创建接口 VPC 终端节点 APIs，可以选择的服务是 com.amazonaws。 ***Region***.stream-logs。要连接 FIPS 终端节点，需要选择的服务是`com.amazonaws.Region.stream-logs-fips`。

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 终端节点的访问权限

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 Logs 的终端节点策略示 CloudWatch 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. 打开位于 https://console.aws.amazon.com/vpc/ 的 Amazon VPC 控制台。
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。然后选择 com.amazonaws。***Region***.logs，然后选择创建终端节点。
4. 选择 com.amazonaws。***Region***.logs 端点，然后选择屏幕下半部分的 “策略” 选项卡。
5. 选择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`aws:SourceVpc`和`aws:SourceVpce`上下文密钥，这些密钥可以限制对特定 VPCs 或特定 VPC 终端节点的访问。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 与一项服务集成 AWS CloudTrail，该服务提供用户、角色或角色所执行操作的记录 AWS 服务。 CloudTrail 将 CloudWatch 日志的 API 调用捕获为事件。捕获的调用包括来自 CloudWatch 日志控制台的调用和对 CloudWatch 日志 API 操作的代码调用。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、发出请求的时间以及其他详细信息。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是用户凭证发出的。
- 请求是否代表 IAM Identity Center 用户发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

CloudTrail 在您创建账户 AWS 账户 时在您的账户中处于活动状态，并且您自动可以访问 CloudTrail 活动历史记录。 CloudTrail 事件历史记录提供了过去 90 天中记录的管理事件的可查看、可搜索、可下载且不可变的记录。 AWS 区域有关更多信息，请参阅《AWS CloudTrail 用户指南》中的 “使用 CloudTrail 事件历史记录”。查看活动历史记录不 CloudTrail收取任何费用。

要持续记录 AWS 账户 过去 90 天内的事件，请创建跟踪或 CloudTrailLake 事件数据存储。

CloudTrail 步道

跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。使用创建的所有跟踪 AWS 管理控制台 都是多区域的。您可以通过使用 AWS CLI创建单区域或多区域跟踪。建议创建多区域跟踪，因为您可以捕获账户 AWS 区域 中的所有活动。如果您创建单区域跟踪，则只能查看跟踪的 AWS 区域中记录的事件。有关跟踪的更多信息，请参阅《AWS CloudTrail 用户指南》中的为您的 AWS 账户创建跟踪和为组织创建跟踪。

通过创建跟踪，您可以免费将正在进行的管理事件的一份副本传送到您的 Amazon S3 存储桶，但会收取 Amazon S3 存储费用。 CloudTrail 有关 CloudTrail 定价的更多信息，请参阅AWS CloudTrail 定价。有关 Amazon S3 定价的信息，请参阅 Amazon S3 定价。

CloudTrail 湖泊事件数据存储

CloudTrail L@@ ak e 允许您对事件运行基于 SQL 的查询。 CloudTrail Lake 将基于行的 JSON 格式的现有事件转换为 Apache ORC 格式。ORC 是一种针对快速检索数据进行优化的列式存储

格式。事件将被聚合到事件数据存储中，它是基于您通过应用高级事件选择器选择的条件的不可变的事件集合。应用于事件数据存储的选择器用于控制哪些事件持续存在并可供您查询。有关 CloudTrail Lake 的更多信息，请参阅AWS CloudTrail 用户指南中的使用 AWS CloudTrail Lake。

CloudTrail 湖泊事件数据存储和查询会产生费用。创建事件数据存储时，您可以选择要用于事件数据存储的定价选项。定价选项决定了摄取和存储事件的成本，以及事件数据存储的默认和最长保留期。有关 CloudTrail 定价的更多信息，请参阅AWS CloudTrail 定价。

CloudWatch 日志支持将以下操作作为事件记录在 CloudTrail 日志文件中：

- AssociateKmsKey
- CancelExportTask
- CreateDelivery
- CreateExportTask
- CreateLogAnomalyDetector
- CreateLogGroup
- CreateLogStream
- DeleteAccountPolicy
- DeleteDataProtectionPolicy
- DeleteDelivery
- DeleteDeliveryDestination
- DeleteDeliveryDestinationPolicy
- DeleteDeliverySource
- DeleteDestination
- DeleteIndexPolicy
- DeleteIntegration
- DeleteLogAnomalyDetector
- DeleteLogGroup
- DeleteLogStream
- DeleteMetricFilter
- DeleteQueryDefinition
- DeleteResourcePolicy

- DeleteRetentionPolicy
- DeleteSubscriptionFilter
- DeleteTransformer
- DescribeAccountPolicies
- DescribeConfigurationTemplates
- DescribeDeliveries
- DescribeDeliveryDestinations
- DescribeDeliverySources
- DescribeDestinations
- DescribeExportTasks
- DescribeFieldIndexes
- DescribeIndexPolicies
- DescribeLogGroups
- DescribeLogStreams
- DescribeMetricFilters
- DescribeQueries
- DescribeQueryDefinitions
- DescribeResourcePolicies
- DescribeSubscriptionFilters
- DisassociateKmsKey
- FilterLogEvents
- GetDataProtectionPolicy
- GetDelivery
- GetDeliveryDestination
- GetDeliveryDestinationPolicy
- GetDeliverySource
- GetIntegration
- GetLogAnomalyDetector
- GetLogEvents
- GetLogGroupFields

- GetLogRecord
- GetQueryResults
- GetTransformer
- ListAnomalies
- ListIntegrations
- ListLogAnomalyDetectors
- ListLogGroups
- ListLogGroupsForQuery
- ListTagsForResource
- ListTagsLogGroup
- PutAccountPolicy
- PutDataProtectionPolicy
- PutDeliveryDestination
- PutDeliveryDestinationPolicy
- PutDeliverySource
- PutDestination
- PutDestinationPolicy
- PutIndexPolicy
- PutIntegration
- PutMetricFilter
- PutQueryDefinition
- PutResourcePolicy
- PutRetentionPolicy
- PutSubscriptionFilter
- PutTransformer
- StartLiveTail
- StartQuery
- StopQuery
- TagResource
- TestMetricFilter

- TestTransformer
- UntagResource
- UpdateAnomaly
- UpdateDeliveryConfiguration
- UpdateLogAnomalyDetector

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 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"
}
```

使用 CloudWatch 指标进行监控

您可以使用本节中的表格来查看 Amazon CloudWatch Logs CloudWatch 每分钟发送给亚马逊的指标。

CloudWatch 记录指标

AWS/Logs 命名空间包括以下指标。

指标	说明
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。

指标	说明
	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
`EMFDisabledErrors`	由于与日志组匹配的 METRIC_EXTRACTION_POLICY 类型的活动账户策略而被忽略的 EMF 格式的日志事件的数量。有关指标提取策略的更多信息，请参阅。PutAccountPolicy 有效维度：`LogGroupName` 有效统计数据：Sum 单位：无
`EMFParsingErrors`	处理嵌入式指标格式日志时遇到的解析错误数量。当日志被识别为嵌入式指标格式但不遵循正确的格式时，就会发生此类错误。有关嵌入式指标格式的更多信息，请参阅规范：嵌入式指标格式。 有效维度：`LogGroupName` 有效统计数据：Sum 单位：无

指标	说明
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
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

指标	说明
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
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

指标	说明
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

下表列出了可用于大多数 CloudWatch Logs 指标的维度。

维度	说明
LogGroupName	要显示其指标的 CloudWatch 日志日志组的名称。
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 (?)。

订阅筛选指标维度

下表列出了与账户级订阅筛选条件相关的指标维度。

维度	说明
PolicyLevel	策略适用的级别。目前，此维度的唯一有效值是 AccountPolicy

维度	说明
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 (?)。

日志转换器指标和维度

CloudWatch Logs 将以下日志转换器指标发布到AWS/Logs命名空间 CloudWatch 中。

指标	说明
TransformationErrors	使用指定转换器转换日志事件时遇到的错误数量。 单位：无 有效统计数据：总计
TransformedBytes	转换后的日志事件的输出量，以未压缩字节为单位。 单位：字节 有效统计数据：总计
TransformedLogEvents	转换后的日志事件数量。 单位：无 有效统计数据：总计

转换器指标会使用以下维度。

维度	说明
LogGroupname	此尺寸仅用于 log-group-level变压器。

维度	说明
PolicyLevel	此维度仅用于账户级转换器。目前，此维度的唯一有效值是 AccountPolicy

集中化指标和维度

要对多个账户和区域进行集中监控，您可以使用 CloudWatch 日志集中将日志数据和指标整合到一个中心位置。有关更多信息，请参阅 Cross-account 跨区域日志集中化。

CloudWatch Logs 将以下集中化指标发布到AWS/Logs命名空间 CloudWatch 中。这些指标可帮助您在使用日志集中化规则时监控从源账户到目标账户的 CloudWatch 日志数据复制情况。

> **Note**
>
> CloudWatch 在您创建集中化规则后不久就开始报告集中化指标。指标是尽力发布的，并且仅跟踪在创建集中规则后到达的新日志事件。

指标	说明	发布位置
IncomingCopiedBytes	复制到目标账户的日志数据量（以未压缩字节为单位）。此指标仅适用于创建集中化规则后到达的新日志事件。 有效尺寸： SourceRegion， SourceAccount 有效统计数据：Sum 单位：字节	目标账户
IncomingCopiedLogEvents	复制到目标账户的日志事件数量。此指标仅适用于创建集中化规则后到达的新日志事件。 有效尺寸： SourceRegion， SourceAccount 有效统计数据：Sum	目标账户

指标	说明	发布位置
	单位：无	
OutgoingCopiedBytes	从源账户发送到目标账户的日志数据量（以未压缩字节为单位）。此指标仅适用于在创建集中规则后复制到目标的新日志事件。 有效尺寸： DestinationRegion 有效统计数据：Sum 单位：字节	源账户
OutgoingCopiedLogEvents	从源账户发送到目标账户的日志事件数量。此指标仅适用于在创建集中规则后复制到目标的新日志事件。 有效尺寸： DestinationRegion 有效统计数据：Sum 单位：无	源账户
CentralizationError	复制日志数据时遇到的错误数。错误可能由多种原因导致，例如 KMS 密钥权限问题、日志组配额限制或日志层不匹配。要识别复制失败的特定日志组或日志流及其失败原因，请监控ErrorType 维度。 有效尺寸： ErrorType 有效统计数据：Sum 单位：无	目标账户

指标	说明	发布位置
CentralizationThrottled	集中化处理被限制的次数。限制会导致集中化处理速度变慢，但不会阻止日志数据的复制。 有效尺寸： DestinationRegion 有效统计数据：Sum 单位：无	源账户

集中化指标使用以下维度。

维度	说明
SourceRegion	源日志数据来源的 AWS 区域。
SourceAccount	源日志数据来源的 AWS 账户 ID。
DestinationRegion	要将日志数据复制到的 AWS 区域。
ErrorType	集中化期间遇到的错误类型。可能的值包括： • LogGroupQuotaExceeded ：目标账户已达到其日志组配额。 • InvalidKMS ：KMS 密钥丢失、已删除、未启用，或者是非对称而不是对称的。 • AccessDenied ：尝试复制日志数据时访问被拒绝。这可能是由于权限不足所致，例如 KMS 密钥权限不正确、缺少 IAM 权限或资源策略受限。 • LogTierMismatch ：源日志组和目标日志组的日志层不匹配。 • InvalidLogStream ：创建或写入日志流时遇到的参数无效，例如当日志流名称超过最大长度时。 • InvalidLogGroup ：在目标账户中创建或配置日志组时遇到的参数无效。

维度	说明
	• DestinationEncryptionMismatch ：目标日志组已存在，其 KMS 加密配置与集中化规则指定的配置不同。

CloudWatch 记录服务使用情况指标

CloudWatch 日志向其发送指标 CloudWatch ，用于跟踪日 CloudWatch 志 API 操作的使用情况。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限额。有关更多信息，请参阅 Service Quotas 集成和使用量指标。

例如，您可以跟踪 ThrottleCount 指标或为该指标设置警报。如果该指标的值上升，则应考虑为受到限制的 API 操作请求增加配额。有关 CloudWatch 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。

Metrics

指标	说明
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。

Dimensions

维度	说明
Account metrics	使用此维度获取所有 CloudWatch 日志中指标的总和 APIs。 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。
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

您可以使用本节中的表格在 Amazon Logs 中查看 AWS 账户的默认服务配额，也称为 CloudWatch 限制。大多数服务配额（但不是全部）都列在 Service Quotas 控制台的 Amazon CloudWatch Logs 命名空间下。

> **Note**
>
> 如果想要请求增加任何这些配额的配额，请参阅本节中的过程。

Name	默认值	可调整	说明
活动导出任务	每个受支持的区域：1 个	否	每个账户的活动（正在运行或待处理）导出任务的数量
AssociateKmsKey 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大关联 kms-key 调用次数 account/per
AssociateSourceToS3TableIntegration 每秒交易次数的节流限制	每个受支持的区域：5 个	否	每个 account/per 区域每秒的最大 AssociateSourceToS3TableIntegration 呼叫数
AssociateSourceToS3TableIntegration 突发事件中每秒交易的节流限制	每个受支持的区域：10 个	否	突发事件中每个 account/per 区域每秒的最大 AssociateSourceToS3TableIntegration 呼叫数
批次大小	每个受支持的区域：1MB	否	put-log-events 请求的最大批处理大小（以 MB 为单位）

Name	默认值	可调整	说明
CancelExportTask 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒取消导出任务调用的最大数量 account/per
CancelImportTask 每秒交易次数的节流限制	每个受支持的区域：1 个	否	每个 account/per 区域每秒的最大 CancelImportTask 呼叫数
CancelImportTask 突发事件中每秒交易的节流限制	每个受支持的区域：1 个	否	突发事件中每个 account/per 区域每秒的最大 CancelImportTask 呼叫数
CreateExportTask 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大创建-导出任务调用数 account/per
CreateImportTask 每秒交易次数的节流限制	每个受支持的区域：1 个	否	每个 account/per 区域每秒的最大 CreateImportTask 呼叫数
CreateImportTask 突发事件中每秒交易的节流限制	每个受支持的区域：1 个	否	突发事件中每个 account/per 区域每秒的最大 CreateImportTask 呼叫数
CreateLogGroup 每秒交易次数的节流限制	每个受支持的区域：每秒 10 个	是	每个区域每秒创建日志组调用的最大数量 account/per
CreateLogStream 每秒交易次数的节流限制	每个受支持的区域：每秒 50 个	是	每个区域每秒创建日志流调用的最大数量 account/per
数据存档	每个受支持的区域：5 GB	否	免费数据存档的最大大小（以 GB 为单位）

Name	默认值	可调整	说明
DeleteDataProtectionPolicy 每秒交易次数的节流限制	每个受支持的区域：5 个	否	每个区域每秒的最大删除数据保护策略调用次数 account/per
DeleteDestination 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大删除目标呼叫数 account/per
DeleteLogGroup 每秒交易次数的节流限制	每个受支持的区域：每秒 10 个	是	每个区域每秒的最大删除日志组调用次数 account/per
DeleteLogStream 每秒交易次数的节流限制	每个受支持的区域：每秒 15 个	是	每个区域每秒的最大删除日志流调用次数 account/per
DeleteMetricFilter 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大删除指标筛选调用次数 account/per
DeleteRetentionPolicy 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大删除保留策略调用次数 account/per
DeleteSubscriptionFilter 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大删除订阅过滤调用次数 account/per
DeleteTransformer 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大删除转换器调用次数 account/per
DescribeDestinations 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大描述目标呼叫数 account/per

Name	默认值	可调整	说明
DescribeExportTasks 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大描述导出任务调用次数 account/per
DescribeImportTaskBatches 每秒交易次数的节流限制	每个受支持的区域：10 个	否	每个 account/per 区域每秒的最大 DescribeImportTaskBatches 呼叫数
DescribeImportTaskBatches 突发事件中每秒交易的节流限制	每个受支持的区域：10 个	否	突发事件中每个 account/per 区域每秒的最大 DescribeImportTaskBatches 呼叫数
DescribeImportTasks 每秒交易次数的节流限制	每个受支持的区域：10 个	否	每个 account/per 区域每秒的最大 DescribeImportTasks 呼叫数
DescribeImportTasks 突发事件中每秒交易的节流限制	每个受支持的区域：10 个	否	突发事件中每个 account/per 区域每秒的最大 DescribeImportTasks 呼叫数
DescribeLogGroups 每秒交易次数的节流限制	每个受支持的区域：每秒 10 个	是	每个区域每秒的最大描述日志组调用次数 account/per
DescribeLogStreams 每秒交易次数的节流限制	每个受支持的区域：每秒 25 个	是	每个区域每秒的最大描述日志流调用次数 account/per
DescribeMetricFilters 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大描述指标过滤器调用次数 account/per

Name	默认值	可调整	说明
DescribeSubscriptionFilters 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大描述-订阅过滤器调用次数 account/per
DisassociateSourceFromS3TableIntegration 每秒交易次数的节流限制	每个受支持的区域：5 个	否	每个 account/per 区域每秒的最大 DisassociateSourceFromS3TableIntegration 呼叫数
DisassociateSourceFromS3TableIntegration 突发事件中每秒交易的节流限制	每个受支持的区域：10 个	否	突发事件中每个 account/per 区域每秒的最大 DisassociateSourceFromS3TableIntegration 呼叫数
事件大小	每个支持的区域：1024 KB	否	最大日志事件大小（以 KB 为单位）

Name	默认值	可调整	说明
FilterLogEvents 每秒交易次数的节流限制	us-east-1：每秒 25 次 ap-northeast-3：每秒 5 次 ap-southeast-3：每秒 5 个 ca-west-1：每秒 5 个 eu-central-1：每秒 5 次 il-central-1：每秒 5 每个其他支持的区域：每秒 10 次	否	每个区域每秒的最大过滤器日志事件调用数 account/per
GetDataProtectionPolicy 每秒交易次数的节流限制	每个受支持的区域：5 个	否	每个区域每秒获取数据保护策略的最大调用次数 account/per

Name	默认值	可调整	说明
GetLogEvents 每秒交易次数的节流限制	us-west-2：每秒 10 次 ap-northeast-3：每秒 10 次 ap-southeast-3：每秒 10 个 ca-west-1：每秒 10 eu-central-1：每秒 10 次 eu-west-1：每秒 10 次 eu-west-3：每秒 30 次 il-central-1：每秒 10 每个其他支持的区域：每秒 25 次	否	每个区域每秒的最大获取日志事件调用次数 account/per
GetQueryResults 每秒交易次数的节流限制	每个受支持的区域：10 个	否	每个区域每秒获取查询结果的最大调用次数 account/per
GetTransformer 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒 get-transformer 调用的最大数量 account/per

Name	默认值	可调整	说明
ListSourcesForS3TableIntegration 每秒交易次数的节流限制	每个受支持的区域：5 个	否	每个 account/per 区域每秒的最大 ListSourcesForS3TableIntegration 呼叫数
ListSourcesForS3TableIntegration 突发事件中每秒交易的节流限制	每个受支持的区域：10 个	否	突发事件中每个 account/per 区域每秒的最大 ListSourcesForS3TableIntegration 呼叫数
ListTagsForResource 每秒交易次数的节流限制	每个受支持的区域：每秒 30 个	否	每个区域每秒资源列表标签调用的最大次数 account/per
ListTagsLogGroup 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒钟列表-标签-日志组调用的最大数量 account/per
Live Tail 会话限制	每个支持的区域：15 个	是	每个账户的并发活动 Live Tail 会话的最大数量
日志组	每个受支持的区域：100 万个	是	一个账户可以拥有的日志组的最大数量
每个 Live Tail 会话扫描的日志组数	每个受支持的区域：10 个	否	每个 Live Tail 会话可以扫描的日志组的最大数量
每个日志组的指标筛选条件	每个受支持的区域：100 个	否	每个日志组的指标筛选条件的数量
PutBearerTokenAuthentication 每秒交易次数的节流限制	每个受支持的区域：5 个	否	每个区域每秒的最大看跌持有者令牌身份验证调用次数 account/per

Name	默认值	可调整	说明
PutBearerTokenAuthentication 突发事件中每秒交易的节流限制	每个受支持的区域：10 个	否	突发事件中每个区域每秒的最大看跌持有者令牌身份验证调用次数 account/per
PutDataProtectionPolicy 每秒交易次数的节流限制	每个受支持的区域：5 个	否	每个区域每秒的最大放置数据保护策略调用次数 account/per
PutDestination 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大放置目标呼叫数 account/per
PutDestinationPolicy 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大放置目标策略调用次数 account/per
PutLogEvents 每秒交易次数的节流限制	每个支持的区域：每秒 5000 个	是	每个区域每秒的最大放置日志事件调用次数 account/per
PutMetricFilter 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒放置指标过滤器调用的最大数量 account/per
PutRetentionPolicy 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大看跌期权策略调用次数 account/per
PutSubscriptionFilter 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大看跌订阅过滤调用次数 account/per
PutTransformer 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大看跌期权变压器调用次数 account/per

Name	默认值	可调整	说明
资源策略	每个受支持的区域：10 个	否	每个 account/per 区域的最大资源策略数量
StartLiveTail 每秒交易次数的节流限制	每个受支持的区域：5 个	否	每个区域每秒的最大起始实时尾部呼叫数。account/per 此限制独立适用于控制台和 API
StartQuery 每秒交易次数的节流限制	每个受支持的区域：10 个	否	每个区域每秒启动查询调用的最大数量 account/per
每个日志组的订阅筛选条件	每个受支持的区域：5 个	否	每个日志组的订阅筛选条件的数量
TagLogGroup 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大标签日志组调用次数 account/per
TagResource 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大标签资源调用次数 account/per
TestMetricFilter 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大测试指标过滤器调用次数 account/per
TestTransformer 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒的最大测试变压器调用次数 account/per
UntagLogGroup 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒取消标记日志组调用的最大数量 account/per

Name	默认值	可调整	说明
UntagResource 每秒交易次数的节流限制	每个受支持的区域：每秒 5 个	否	每个区域每秒取消标记资源调用的最大次数 account/per
日志异常检测器数	每个受支持的区域：500 个	是	活动日志异常检测器的最大数量

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《服务配额用户指南》中的什么是服务配额？。

Service Quotas 可以轻松查询 CloudWatch 日志服务配额的值。

AWS 管理控制台

使用控制台查看 CloudWatch 日志服务配额

1. 在 https://console.aws.amazon.com/servicequotas/ 打开 Service Quotas 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。

   在服务配额列表中，您可以查看服务配额名称、应用的值（如果该值可用）、 AWS 默认配额以及配额值是否可调整。
4. 要查看有关服务配额的其他信息（如描述），请选择配额名称。
5. （可选）要请求增加配额，请选择要增加的配额，选择 Request quota increase（请求增加配额），输入或选择所需信息，然后选择 Request（请求）。

要使用控制台进一步处理服务配额，请参阅 Service Quotas 用户指南。要请求提高配额，请参阅 Service Quotas 用户指南中的请求增加配额。

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

下表描述了自 2018 年 6 月起《 CloudWatch 日志用户指南》每个版本的重要更改。要获得本文档的更新通知，您可以订阅 RSS 源。

变更	说明	日期
L CloudWatch ogs Insights 的新估算查询命令	现在，您可以使用 L CloudWatch ogs Insights 中的`estimate`命令返回查询将在所选日志组和时间范围内扫描的估计字节数，而无需运行查询。该`estimate`命令不会产生 CloudWatch 日志见解查询费用。有关更多信息，请参阅CloudWatch 日志见解语言查询语法。	2026年7月22日
更新了 AWSServiceRoleForLogDelivery 服务相关角色政策	CloudWatch 添加的日志`kms:GenerateDataKey` 和 AWSServiceRoleForLogDelivery 服务相关角色策略的`kms:Decrypt` 权限。这些权限允许将日志传送到使用服务器端加密和客户管理密钥的 Firehose 交付流 () SSE-CMK。有关更多信息，请参阅 CloudWatch 记录 AWS 服务关联角色的更新。	2026 年 5 月 15 日
保存了带有 CloudWatch 日志见解参数的查询	现在，您可以在 L CloudWatch ogs Insights 中使用命名参数创建可重复使用的查询模板。定义单个模板并在运行时传递不同的值，而不是维护多个几乎相同的已保存查询。有关更多	2026 年 3 月 26 日

	信息，请参阅使用带参数的已保存查询。	
新 CloudWatch 日志管理策略 CloudWatchLogsAPIKeyAccess	CloudWatch Logs 发布了一项新的托管策略`CloudWatchLogsAPIKeyAccess`，该策略启用 CloudWatch Logs API 密钥身份验证和加密日志提取。有关信息，请参阅CloudWatch 记录 AWS 托管策略的更新。	2026 年 2 月 17 日
更新的 CloudWatchLogsReadOnlyAccess 政策	添加了可观测性管理操作的权限，CloudWatchLogsReadOnlyAccess以允许对遥测管道和 S3 表集成进行只读访问。	2025 年 12 月 2 日
更新的 CloudWatchLogsFullAccess 政策	添加了可观测性管理操作的权限，CloudWatchLogsFullAccess以允许对遥测管道和 S3 表集成进行只读访问。	2025 年 12 月 2 日
日志中的 CloudWatch 新日志管理功能	CloudWatch 将日志管理整合到具有内置监管功能的单一服务中，无需在不同的工具和数据存储中存储和维护相同数据的多个副本。有关更多信息，请参阅日志管理。	2025 年 12 月 2 日
日志中的新数据摄取和标准化 CloudWatch 功能	CloudWatch 通过与 AWS 组织集成以及为第三方来源预建的连接器，自动收集跨账户和 AWS 区域的 AWS 销售日志。有关更多信息，请参阅在摄取期间转换日志。	2025 年 12 月 2 日

L CloudWatch ogs Insights 中新增了基于数据源的分析和查询功能	CloudWatch Insights 引入了查询的各个方面。Facet 对于分析日志很有用，因为它们允许您在不运行查询的情况下以交互方式筛选和分组数据。分面是您的日志（例如 ServiceName 或 StatusCode）中的一个字段，用于对日志组进行过滤、聚合和分析。有关更多信息，请参阅使用 facet 对日志进行分组和浏览。	2025 年 12 月 2 日
新的 S3 表集成	与的 S3 表格集成 CloudWatch 允许您 CloudWatch 使用分析引擎（例如亚马逊雅典娜、亚马逊 Redshift 和支持连接到 Apache Iceberg-compatible 商店的第三方工具）访问采集到的日志数据。有关更多信息，请参阅使用 S3 表集成访问日志。	2025 年 12 月 2 日
CloudWatch 日志增加了对 Cross-account 跨区域日志集中化的支持	CloudWatch Logs 增加了对Cross-account 跨区域日志集中化的支持，使 CloudWatch 日志能够将来自 AWS Organizations 成员账户的日志数据收集到中央账户中进行分析。有关更多信息，请参阅Cross-account跨区域日志集中化。	2025 年 9 月 17 日

CloudWatch 日志异常检测已更新示例策略	CloudWatch 日志更新了 KMS 示例策略，通过为和添加条件来缩小其范围`aws:SourceAccount` 并提高安全性`aws:SourceArn` 。有关更多信息，请参阅使用 KMS 加密异常检测器及其结果 AWS	2025 年 7 月 14 日
CloudWatch Logs Insights 增加了对亚马逊 VPC 路由服务器日志的支持	CloudWatch Logs Insights 增加了对亚马逊 VPC 路由服务器日志的支持。有关更多信息，请参阅启用从 AWS 服务进行日志记录。Amazon VPC 路由服务器的新日志源记录`EVENT_LOGS` 在 PutDeliverySource	2025 年 6 月 12 日
CloudWatch Logs Insights 增加了对 AWS PCS 日志的支持	CloudWatch Logs Insights 增加了对 AWS PCS 日志的支持。有关更多信息，请参阅启用从 AWS 服务进行日志记录。`PCS_SCHEDULER_LOGS` 和的新日志源记录`PCS_JOBCOMP_LOGS` 在 AWS PCS PutDeliverySource	2025 年 6 月 12 日
CloudWatch Logs Insights 增加了对 AWS Entity Resolution 数据匹配服务 日志的支持	CloudWatch Logs Insights 增加了对 AWS Entity Resolution 数据匹配服务 日志的支持。有关更多信息，请参阅启用从 AWS 服务进行日志记录。的新日志源记录`WORKFLOW_LOGS` 在 AWS Entity Resolution 数据匹配服务 PutDeliverySource	2025 年 5 月 22 日

CloudWatch 记录托管策略更新以支持自然语言摘要	的权限`cloudwatch:GenerateQueryResultsSummary` 已添加到CloudWatchLogsFullAccess查询结果中，CloudWatchLogsReadOnlyAccess允许生成查询结果的自然语言摘要。要查看策略的内容，请参阅AWS 托管策略参考指南CloudWatchLogsReadOnlyAccess中的CloudWatchLogsFullAccess和。	2025 年 5 月 20 日
CloudWatch Logs Insights 增加了对从 L CloudWatch ogs Insights 查询结果生成自然语言摘要的支持	在 L CloudWatch ogs Insights 中添加了对自然语言摘要的支持。此功能会生成用户可读的查询结果摘要，目前在美国东部（弗吉尼亚州北部）可用。有关更多信息，请参阅根据 L CloudWatch ogs Insights 查询结果生成自然语言摘要。	2025 年 5 月 20 日

CloudWatchOpenSearchDashboardsAccessCloudWatchOpenSearchDashboardsFullAccess还有新的 IAM 政策	CloudWatch 日志添加了两个新的 IAM 策略，CloudWatchOpenSearchDashboardsFullAccess和CloudWatchOpenSearchDashboardsAccess。CloudWatchOpenSearchDashboardsFullAccess授予创建和管理与 S OpenSearch ervice 集成的权限。CloudWatchOpenSearchDashboardsAccess授予查看在这些集成中创建的供应商日志仪表板的权限。	2024 年 12 月 1 日
CloudWatchLogsFullAccess政策已更新	CloudWatch 日志为CloudWatchLogsFullAccess策略添加了权限 Amazon OpenSearch Service 和 IAM，以允许 CloudWatch 日志与 OpenSearch 服务集成以使用某些功能。	2024 年 12 月 1 日
CloudWatch Logs Insights 为查询语法添加了新的结构类型	CloudWatch Logs Insights 添加了`unnest`命令和两个 JSON 函数，使您能够将 JSON 字符串作为地图和列表进行操作。有关更多信息，请参阅结构类型。	2024 年 11 月 21 日
CloudWatch 日志支持在日志摄取期间进行日志转换	您可以创建日志转换器来在摄取时修改日志事件，从而帮助您将不同格式和来自不同来源的日志标准化为一致且上下文丰富的格式。有关更多信息，请参阅在摄取期间转换日志。	2024 年 11 月 20 日

CloudWatch 日志见解增加了字段索引	CloudWatch Logs Insights 增加了对日志字段索引的支持。当您随后在 L CloudWatch ogs Insights 查询中使用字段索引时，该查询会尝试跳过处理已知不包含索引字段的日志事件。有关更多信息，请参阅创建字段索引以提高查询性能和减少扫描量。	2024 年 11 月 20 日
CloudWatch Logs Insights 通常支持自然语言查询生成	CloudWatch Logs Insights 支持自然语言来生成和更新查询。有关更多信息，请参阅使用自然语言生成和更新 CloudWatch 日志见解查询。	2024 年 6 月 20 日
CloudWatchLogsReadOnlyAccess政策已更新	CloudWatch 日志向添加了`cloudwatch:GenerateQuery` 权限 CloudWatchLogsReadOnlyAccess，因此使用此策略的用户可以根据自然语言提示生成 L CloudWatch ogs Insights 查询字符串。	2023 年 11 月 26 日
CloudWatchLogsFullAccess政策已更新	CloudWatch 日志向添加了`cloudwatch:GenerateQuery` 权限 CloudWatchLogsFullAccess，因此使用此策略的用户可以根据自然语言提示生成 L CloudWatch ogs 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，并发现日志数据中的异常。有关更多信息，请参阅日志异常检测。	2023 年 11 月 26 日
CloudWatch 日志添加了比较功能	现在，您可以使用 CloudWatch 日志见解来比较日志事件随时间推移而发生的变化。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，您还可以使用功能齐全的选项。有关更多信息，请参阅日志类。	2023 年 11 月 26 日
CloudWatch Logs Insights 支持自然语言查询生成	CloudWatch Logs Insights 支持自然语言来生成和更新查询。有关更多信息，请参阅使用自然语言生成和更新 CloudWatch 日志见解查询。	2023 年 11 月 26 日
CloudWatch Logs 为 Live Tail 添加了正则表达式过滤器模式语法支持	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 L CloudWatch ogs 用户指南中的筛选模式语法。	2023 年 11 月 13 日

CloudWatch 日志为指标筛选器、订阅过滤器和过滤器日志事件添加了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 L CloudWatch ogs 用户指南中的筛选模式语法。	2023 年 9 月 5 日
CloudWatch Logs Insights 添加了模式命令	现在，您可以在 L CloudWatch ogs 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《亚马逊 CloudWatch 日志用户指南》中的模式。	2023 年 7 月 17 日
CloudWatch Logs Insights 添加了重复数据删除命令	现在，您可以在 L CloudWatch ogs Insights 查询中使用重复数据删除功能，根据指定字段中的特定值删除重复结果。有关更多信息，请参阅《亚马逊 CloudWatch 日志用户指南》中的重复数据删除。	2023 年 6 月 20 日
Account-level 数据保护政策	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，因此您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 日志向添加了权限CloudWatchLogsReadOnlyAccess。添加`logs:StartLiveTail` 和权`logs:StopLiveTail` 限是为了使用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
CloudWatch 日志见解已发布	您可以使用 CloudWatch 日志见解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《亚马逊日志用户指南》中的使用 CloudWatch 日志见解分析 CloudWatch 日志数据	2018 年 11 月 27 日
对 Amazon VPC 端点的支持	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 Logs 用户指南中的将 CloudWatch 日志与接口 VPC 终端节点结合使用。	2018 年 6 月 28 日

下表描述了《亚马逊 CloudWatch 日志用户指南》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亚马逊 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 将 CloudWatch 日志与接口 VPC 终端节点配合使用。	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
增加了对亚马逊 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 Amazon S3 的支持	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
增加了对亚马逊 CloudWatch 日志中 AWS CloudTrail 记录的事件的支持	您可以在捕获的特定 API 活动中创建警报 CloudWatch 和接收通知， CloudTrail 并使用该通知进行故障排除。	2014 年 11 月 10 日
增加了对亚马逊 CloudWatch 日志的支持	您可以使用亚马逊 CloudWatch 日志监控、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，您可以使用亚马逊 CloudWatch 控制台、中的 CloudWatch 日志命令或日志软件开发工具包从 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。