Amazon CloudWatch Logs

用户指南

Amazon CloudWatch Logs: 用户指南

Table of Contents

什么是 Amazon CloudWatch Logs？

您可以使用 Amazon CloudWatch Logs 来监控、存储和访问来自 Amazon Elastic Compute Cloud (Amazon EC2) 实例、AWS CloudTrail、Route 53 和其他来源的日志文件。您随后可以从 CloudWatch Logs 中检索关联的日志数据。

特色

- 实时监控来自 Amazon EC2 实例的日志 - 您可以使用 CloudWatch Logs 通过日志数据监控应用程序和系统。例如，CloudWatch Logs 能够跟踪应用程序日志中的错误数，并在错误率超过指定阈值时向您发送通知。CloudWatch Logs 使用您的日志数据进行监控，因此无需更改代码。例如，您可以监控应用程序日志以查找特定字词 (如“NullReferenceException”) 或日志数据中特定位置处某个字词 (如 Apache 访问日志中的“404”状态代码) 出现的次数。找到您要搜索的字词时，CloudWatch Logs 向您指定的 CloudWatch 指标报告该数据。日志数据会在传输期间加密，并且会对静态日志数据加密。要了解其用法，请参阅 CloudWatch Logs 入门 (p. 5)。
- 监控 AWS CloudTrail 记录的事件 – 您可以在 CloudWatch 中创建警报并接收 CloudTrail 捕获的特定 API 活动的通知，然后使用通知执行问题排查。在开始时，请参阅 AWS CloudTrail User Guide 中的将 CloudTrail 事件发送到 CloudWatch Logs。
- 归档日志数据 - 您可以使用 CloudWatch Logs 在高持久性存储中存储日志数据。您可以更改日志保留设置，以便自动删除存在时间超过此设置的所有日志事件。CloudWatch Logs 代理支持您轻松快速地将已轮换和未轮换的日志文件从主机移动到日志服务。然后，您可以按需访问原始日志数据。
- 记录 Route 53 DNS 查询—您可以使用 CloudWatch Logs 记录有关 Route 53 收到的 DNS 查询的信息。有关更多信息，请参阅 Amazon Route 53 开发人员指南 中的记录 DNS 查询。

相关 AWS 服务

以下服务可与 CloudWatch Logs 一起使用：

- AWS CloudTrail 是一项 Web 服务，可用来监控对账户的 CloudWatch Logs API 进行的调用，包括由 AWS 管理控制台、命令行界面 (CLI) 和其他服务发出的调用。启用 CloudTrail 日志记录后，CloudTrail 将捕获您的帐户中的 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 AWS CloudTrail 的更多信息，请参阅 AWS CloudTrail User Guide 中的什么是 AWS CloudTrail？。有关由 CloudWatch 写入 CloudTrail 日志文件的数据类型的示例，请参阅在 AWS CloudTrail 中记录 Amazon CloudWatch Logs API 调用 (p. 87)。
- 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。通过 IAM 可以控制哪些人可以使用您的 AWS 资源 (身份验证) 以及他们可以使用的资源和采用的方式 (授权)。有关更多信息，请参阅什么是 IAM？(在 IAM 用户指南 中)。
- Amazon Kinesis Data Streams 是一项 Web 服务，可用于实现快速而持续的数据引入和聚合。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，请参阅什么是 Amazon Kinesis Data Streams？(在 Amazon Kinesis Data Streams 开发人员指南 中)。
- 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、容量预配置和自动扩展、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 Developer Guide 中)。

定价

注册 AWS 后，您可以通过 AWS 免费套餐开始免费使用 CloudWatch Logs。

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

有关更多信息，请参阅 Amazon CloudWatch 定价。

Amazon CloudWatch Logs 概念

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

日志事件

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

日志流

日志流是共享同一个源的一系列日志事件。具体来说，日志流通常用于表示受监视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 访问日志关联。当您不再需要日志流时，可使用 aws logs delete-log-stream 命令将其删除。此外，AWS 可能会删除 2 个月以前的空日志流。

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。

指标筛选器

指标筛选器可用于表示服务如何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。指标筛选器将分配给日志组，分配给日志组的所有筛选器都将应用于其日志流。

保留期设置

保留期设置可用于指定日志事件保留在 CloudWatch Logs 中多长时间。过期的日志事件将自动删除。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。

CloudWatch Logs 限制

CloudWatch Logs 有以下限制：

资源	默认限制
批处理大小	1 MB (最大)。此限制不能更改。
数据存档	最多免费存档 5GB 的数据。此限制不能更改。
DescribeLogStreams	每秒 5 个事务 (TPS/账户/区域)。 如果遇到常见限制，您可以请求提高限制。
事件大小	256 KB (最大)。此限制不能更改。

资源	默认限制
导出任务	每个账户一次一个活动 (运行中或等待中) 的导出任务。此限制不能更改。
FilterLogEvents	每秒 5 个事务 (TPS)/账户/区域. 只能在特殊情况下更改此限制。如果您遇到常见限制，请联系 AWS Support。有关说明，请参阅 AWS 服务限制。
GetLogEvents	每个区域的每个账户每秒 10 个请求。如果您持续处理新的数据，建议您进行订阅。如果您需要历史数据，建议将您的数据导出到 Amazon S3。 只能在特殊情况下更改此限制。如果您遇到常见限制，请联系 AWS Support。有关说明，请参阅 AWS 服务限制。
传入数据	最多免费传入 5GB 的数据。此限制不能更改。
日志组	每个区域的每个账户 5000 个日志组。您可以请求提高限制。 对可属于一个日志组的日志流数没有限制。
指标筛选器	每个日志组 100 个。此限制不能更改。
PutLogEvents	每个日志流每秒 5 个请求。额外的请求将被阻止。此限制不能更改。 PutLogEvents 请求的最大批处理大小为 1MB。 每区域每账户每秒 800 个事务。您可以请求提高限制。
订阅筛选器	每个日志组 1 个。此限制不能更改。

开始设置

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

注册 Amazon Web Services (AWS)

创建 AWS 账户时，我们会自动为所有 AWS 服务注册您的账户。您只需为使用的服务付费。

如果您已有一个 AWS 账户，请跳到下一个步骤。如果您还没有 AWS 账户，请使用以下步骤创建。

如需注册 AWS 账户

1. 打开 http://amazonaws.cn/，然后选择 Create an AWS Account。

   Note

   如果您之前已登录 AWS 管理控制台，则可能无法在浏览器中执行此操作。在此情况下，请选择 Sign in to a different account，然后选择 Create a new AWS account。
2. 按照屏幕上的说明进行操作。

   作为注册流程的一部分，您会收到一个电话，需要您使用电话键盘输入一个 PIN 码。

登录 Amazon CloudWatch 控制台

登录 Amazon CloudWatch 控制台的步骤

1. 登录 AWS 管理控制台并通过以下网址打开 CloudWatch 控制台 https://console.amazonaws.cn/cloudwatch/。
2. 如果需要，可以更改区域。从导航栏中，选择 AWS 资源所在的区域。
3. 在导航窗格中，选择 Logs。

设置命令行界面

您可以使用 AWS CLI 执行 CloudWatch Logs 操作。

有关如何安装和配置 AWS CLI 的信息，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。

CloudWatch Logs 入门

您可以从运行 Linux 或 Windows Server 的 Amazon EC2 实例发布日志数据，以及从 AWS CloudTrail 发布记录的事件。CloudWatch Logs 可使用来自任何区域中的资源的日志，但是您只能在支持 CloudWatch Logs 的区域的 CloudWatch 控制台中查看日志数据。有关更多信息，请参阅 Amazon Web Services 一般参考 中的区域和终端节点。

要在运行 Microsoft Windows 的 EC2 实例上开始使用 CloudWatch Logs，请参阅 Amazon EC2 用户指南（适用于 Windows 实例）中的将性能计数器发送到 CloudWatch 并将日志发送到 CloudWatch 日志。

要开始使用 CloudWatch Logs 和 CloudTrail 中记录的事件，请参阅 AWS CloudTrail User Guide 中的将 CloudTrail 事件发送到 CloudWatch Logs。

内容

CloudWatch Logs 代理先决条件

CloudWatch Logs 代理需要 Python 版本 2.6、2.7、3.0 或 3.3，以及以下任何版本的 Linux：

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

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

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

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

步骤 1：为 CloudWatch Logs 配置 IAM 角色或用户

CloudWatch Logs 代理支持 IAM 角色和用户。如果您的实例已有一个与之关联的 IAM 角色，请确保包含下面的 IAM 策略。如果您尚未将 IAM 角色分配给实例，则可以将 IAM 凭证用于后续步骤，也可以向该实例分配 IAM 角色。有关更多信息，请参阅将 IAM 角色附加到实例。

为 CloudWatch Logs 配置 IAM 角色或用户

1. 通过以下网址打开 IAM 控制台：https://console.amazonaws.cn/iam/。
2. 在导航窗格中，选择 Roles。
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. 在 Permissions 选项卡上，展开 Inline Policies，然后选择用于创建内联策略的链接。
5. 在 Set Permissions 页面上，选择 Custom Policy 和 Select。

   有关创建自定义策略的更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的 Amazon EC2 的 IAM 策略。
6. 在 Review Policy 页面上，为 Policy Name 键入策略的名称。
7. 对于 Policy Document，粘贴以下策略：

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

8. 选择 Apply Policy。

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

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

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

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

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

1. 连接到您的 Amazon Linux 实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的连接到您的实例。

   如果在连接时出现问题，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的排除连接实例的故障。
2. 更新您的 Amazon Linux 实例以在软件包存储库中选取最新更改。

   ```
   sudo yum update -y
   ```
3. 安装 `awslogs` 软件包。这是在 Amazon Linux 实例上安装 awslogs 的推荐方法。

   ```
   sudo yum install -y awslogs
   ```
4. 编辑 `/etc/awslogs/awslogs.conf` 文件以配置要跟踪的日志。有关编辑此文件的更多信息，请参阅 CloudWatch Logs 代理参考 (p. 90)。
5. 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
   ```
6. (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。
7. (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，请使用以下命令启动 `awslogs`。

   ```
   sudo systemctl enable awslogsd.service
   ```
8. 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。

   要查看日志，请参阅 查看发送到 CloudWatch Logs 的日志数据 (p. 30)。

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

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

1. 连接到您的 EC2 实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的连接到您的实例。

   如果在连接时出现问题，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的排除连接实例的故障。
2. 使用两个选项之一运行 CloudWatch Logs 代理安装程序。您可以直接从 Internet 运行，也可以下载文件并独立运行。

   Note

   如果您运行的是 CentOS 6.x 或 Red Hat 6.x，请使用相应的步骤下载和运行单独的安装程序，而不是直接从 Internet 运行它。

   Note

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

   要直接从 Internet 运行，请使用以下命令并按照提示操作：

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

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

   如果上述命令不起作用，请尝试以下命令：

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

   要下载并独立运行它，请使用以下命令并按照提示操作：

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

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

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

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

   您可以通过指定 us-east-1, us-west-1, us-west-2, ap-south-1, ap-northeast-2, ap-southeast-1, ap-southeast-2, ap-northeast-1, eu-central-1, eu-west-1, or sa-east-1 区域来安装 CloudWatch Logs 代理。

   Note

   有关 `awslogs-agent-setup` 的当前版本和版本历史记录的详细信息，请参阅 CHANGELOG.txt。

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

Item	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 秘密访问密钥。
默认区域名称	按 Enter。默认值为 us-west-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, or sa-east-1。
默认输出格式	保留空白并按 Enter。
要上传的日志文件的路径	包含待发送日志数据的文件的位置。安装程序将为您提供路径建议。
目标日志组名称	日志组的名称。安装程序将为您提供日志组名称建议。
目标日志流名称	默认情况下，这是主机名称。安装程序将为您提供主机名称建议。
时间戳格式	指定在指定日志文件中使用的时间戳的格式。选择自定义可指定您自己的格式。
初始位置	数据的上传方式。设置为 start_of_file 将上传数据文件中的所有内容。设置为 end_of_file 将仅上传最近追加的数据。

完成这些步骤后，安装程序会询问您是否需要配置另一个日志文件。对于每个日志文件，您都可以运行任意次此过程。如果没有其他要监控的日志文件，当安装程序提示设置其他日志时，请选择 N (否)。有关代理配置文件中的设置的更多信息，请参阅 CloudWatch Logs 代理参考 (p. 90)。

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。

   要查看日志，请参阅 查看发送到 CloudWatch Logs 的日志数据 (p. 30)。

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

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

请注意，不支持配置多个日志源将数据发送到单个日志流。

先决条件

创建用于描述所有日志组和日志流的代理配置文件。这是一个文本文件，描述要监控的日志文件以及要将这些文件上传到的日志组和日志流。代理使用此配置文件并开始监控并上传其中描述的所有日志文件。有关代理配置文件中的设置的更多信息，请参阅 CloudWatch Logs 代理参考 (p. 90)。

以下是适用于 Amazon Linux 的一个示例代理配置文件

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

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

以下是适用于 Ubuntu 的一个示例代理配置文件

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

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

配置您的 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.amazonaws.cn/iam/。
2. 在导航窗格中，选择 Policies 和 Create Policy。
3. 在 Create Policy 页面上，对于 Create Your Own Policy，选择 Select。有关创建自定义策略的更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的 Amazon EC2 的 IAM 策略。
4. 在 Review Policy 页面上，为 Policy Name 键入策略的名称。
5. 对于 Policy Document，粘贴以下策略：

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

6. 选择 Create Policy。
7. 在导航窗格中，选择 Roles 和 Create New Role。
8. 在 Set Role Name 页面上，请键入角色名称，然后选择 Next Step。
9. 在 Select Role Type 页面上，选择 Amazon EC2 旁的 Select。
10. 在 Attach Policy 页面上的表标题中，依次选择 Policy Type 和 Customer Managed。

11. 选择您创建的 IAM 策略，然后选择 Next Step。
12. 选择 Create Role。

    有关 IAM 用户和策略的更多信息，请参阅IAM 用户指南 中的 IAM 用户和组和管理 IAM 策略。

启动新实例和启用 CloudWatch Logs

1. 打开 Amazon EC2 控制台 https://console.amazonaws.cn/ec2/。
2. 选择 Launch Instance。

   有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的启动实例。
3. 在 Step 1: Choose an Amazon Machine Image (AMI) 页面上，选择要启动的 Linux 实例类型，然后在 Step 2: Choose an Instance Type 页面上，选择 Next: Configure Instance Details。

   确保 Amazon 系统映像 (AMI) 中包含了 cloud-init。Amazon Linux AMI 和适用于 Ubuntu 和 RHEL 的 AMI 已包含 cloud-init，但 AWS Marketplace 中的 CentOS 和其他 AMI 可能未包含。
4. 在 Step 3: Configure Instance Details 页面上，为 IAM role 选择您创建的 IAM 角色。
5. 在 Advanced Details 下，将以下脚本粘贴到 User data 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://myawsbucket/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。
7. 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。

   要查看日志，请参阅 查看发送到 CloudWatch Logs 的日志数据 (p. 30)。

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

您可以安装 CloudWatch Logs 代理，使用 AWS OpsWorks 和 Chef 创建流日志，Chef 是第三方系统和云基础设施自动化工具。Chef 使用“配方” (编写的“配方”用于在计算机中安装和配置软件) 和“说明书”(配方的集合) 来执行其配置和策略分配任务。有关更多信息，请参阅 Chef。

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

步骤 1：创建自定义配方

创建存储库来存储配方。AWS OpsWorks 支持 Git 和 Subversion，您也可以在 Amazon S3 中存储存档。AWS OpsWorks 用户指南 中的说明书存储库介绍了说明书存储库结构。以下示例假设说明书名为 `logs`。install.rb 配方可安装 CloudWatch Logs 代理。您也可以下载食谱示例 (CloudWatchLogs-Cookbooks.zip)。

创建包含以下代码、名为 metadata.rb 的文件：

```
#metadata.rb

name             'logs'
```

```
version          '0.0.1'
```

创建 CloudWatch Logs 配置文件：

```
#config.rb

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

下载并安装 CloudWatch Logs 代理：

```
# install.rb

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

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

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

> **Note**
>
> 在上面的示例中，将 *region* 替换为以下区域之一：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, or sa-east-1。

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

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

此配方使用 cwlogs.cfg.erb 模板文件，您可以修改该文件以指定不同属性，如要对哪些文件记录日志。有关这些属性的更多信息，请参阅 CloudWatch Logs 代理参考 (p. 90)。

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


## Each log file is defined in its own section. The section name doesn't
## matter as long as its unique within this file.
#
#[kern.log]
#
## Path of log file for the agent to monitor and upload.
#
#file = /var/log/kern.log
```

```
#
## Name of the destination log group.
#
#log_group_name = kern.log
#
## Name of the destination log stream.
#
#log_stream_name = {instance_id}
#
## Format specifier for timestamp parsing.
#
#datetime_format = %b %d %H:%M:%S
#
#

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

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

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

步骤 2：创建 AWS OpsWorks 堆栈

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

步骤 3：扩展您的 IAM 角色

要将 CloudWatch Logs 用于 AWS OpsWorks 实例，需要扩展实例所用的 IAM 角色。

1. 通过以下网址打开 IAM 控制台：https://console.amazonaws.cn/iam/。
2. 在导航窗格中，选择 Policies 和 Create Policy。
3. 在 Create Policy 页面上的 Create Your Own Policy 下，选择 Select。有关创建自定义策略的更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的 Amazon EC2 的 IAM 策略。
4. 在 Review Policy 页面上，为 Policy Name 键入策略的名称。
5. 对于 Policy Document，粘贴以下策略：

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

```
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
    ],
    "Resource": [
      "arn:aws-cn:logs:*:*:*"
    ]
  }
 ]
}
```

6. 选择 Create Policy。
7. 在导航窗格中，选择 Roles，然后在内容窗格中，为 Role Name 选择您的 AWS OpsWorks 堆栈使用的实例角色的名称。在堆栈设置中，可以看到堆栈使用的角色 (默认角色为 `aws-opsworks-ec2-role`。)

   Note

   选择角色名称，而不是选中复选框。
8. 在 Permissions 选项卡上的 Managed Policies 下，选择 Attach Policy。
9. 在 Attach Policy 页面上的表标题 (Filter 和 Search 旁边) 中，选择 Policy Type 和 Customer Managed Policies。
10. 对于 Customer Managed Policies，选择您在上面创建的 IAM 策略，然后选择 Attach Policy。

    有关 IAM 用户和策略的更多信息，请参阅IAM 用户指南 中的 IAM 用户和组和管理 IAM 策略。

步骤 4：添加层

1. 通过以下网址打开 AWS OpsWorks 控制台：https://console.amazonaws.cn/opsworks/。
2. 在导航窗格中，选择 Layers。
3. 在内容窗格中，选择一个层，然后选择 Add layer。
4. 在 OpsWorks 选项卡上，为 Layer type 选择 Custom。
5. 对于 Name 和 Short name 字段，输入层的长名称和短名称，然后选择 Add layer。
6. Recipes 选项卡的 Custom Chef Recipes 下有多个标题 - Setup、Configure、Deploy、Undeploy 和 Shutdown，它们对应于 AWS OpsWorks 生命周期事件。AWS OpsWorks 在实例生命周期的这些关键点触发这些事件，这些事件运行关联的配方。

   Note

   如果上述标题不可见，请在 Custom Chef Recipes 下，选择 edit。
7. 在 Setup 旁输入 logs::config, logs::install，选择 + 以将它添加到列表中，然后选择 Save。

   实例启动之后，AWS OpsWorks 即在此层的每个新实例上运行此配方。

步骤 5：添加实例

层仅控制如何配置实例。现在您需要将一些实例添加到层并启动它们。

1. 通过以下网址打开 AWS OpsWorks 控制台：https://console.amazonaws.cn/opsworks/。
2. 在导航窗格中，选择 Instances，然后在您的层下选择 + Instance。
3. 接受默认设置，然后选择 Add Instance 以将该实例添加到层。

4. 在该行的 Actions (操作) 列中，单击 start (启动) 以启动该实例。

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

步骤 6：查看您的日志

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。

要查看日志，请参阅 查看发送到 CloudWatch Logs 的日志数据 (p. 30)。

快速入门：使用 AWS CloudFormation 将日志数据发送至 CloudWatch Logs

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

使用 AWS CloudFormation 预配置 AWS 时，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 AWS CloudFormation 设置更丰富的 CloudWatch Logs 部署。有关模板示例，请参阅 AWS CloudFormation 用户指南 中的 Amazon CloudWatch Logs 模板代码段。有关更多入门信息，请参阅 AWS CloudFormation 用户指南 中的 AWS CloudFormation 入门。

快速入门：让您的运行 Windows Server 2016 的 Amazon EC2 实例将日志发送到 CloudWatch Logs

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

步骤

下载示例配置文件

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

为 CloudWatch 配置 JSON 文件

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

步骤

步骤 1：启用 CloudWatch Logs

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

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

步骤 2：为 CloudWatch 配置设置

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

配置设置将日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 `CloudWatchLogs` 部分。

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

2. 将 AccessKey 和 SecretKey 字段留空。您将使用 IAM 角色配置凭证。
3. 对于 Region，键入要将日志数据发送到的区域 (例如 us-east-2)。
4. 对于 LogGroup，键入您的日志组的名称。此名称会显示在 CloudWatch 控制台的 Log Groups 屏幕上。
5. 对于 LogStream，键入目标日志流。此名称会显示在 CloudWatch 控制台的 Log Groups > Streams 屏幕上。

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

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

步骤 3：配置要发送的数据

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

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

1. 在 JSON 文件中，找到 ApplicationEventLog 部分。

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

2. 对于 Levels，指定要上传的消息类型。可以指定以下值之一：

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

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

将安全日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 SecurityEventLog 部分。

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

2. 对于 Levels，键入 **7** 以上传所有消息。

将系统事件日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 `SystemEventLog` 部分。

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

2. 对于 `Levels`，指定要上传的消息类型。可以指定以下值之一：

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

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

将其他类型的事件日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，添加新部分。每个部分必须有一个独立的 `Id`。

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

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

   a. 打开事件查看器。

   b. 在导航窗格中，选择 Applications and Services Logs。

   c. 导航到该日志，然后选择 Actions、Properties。
4. 对于 `Levels`，指定要上传的消息类型。可以指定以下值之一：

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

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

将 Windows 事件跟踪数据发送到 CloudWatch Logs

ETW (Windows 事件跟踪) 提供了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`，请键入您要使用的时间戳格式。有关支持的值的列表，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。有关支持的值的列表，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。有关支持的值的列表，请参阅 MSDN 上产品行为主题中表格中的 `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。如果此参数保留空白且时间戳不包括时区信息，则 CloudWatch Logs 默认为本地时区。如果时间戳已包含时区信息，则忽略此参数。
8. (可选) 对于 `LineCount`，在标头中键入行数以识别日志文件。例如，IIS 日志文件拥有几乎相同的标头。您可以输入 **5**，系统会读取日志文件标头的前三行以进行识别。在 IIS 日志文件中，第三行为日期和时间戳，但无法始终保证时间戳在日志文件之间是不同的。为此，建议包含至少一行实际日志数据以便对日志文件进行唯一指纹识别。

将 IIS 日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 `IISLog` 部分。

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

2. 对于 `LogDirectoryPath`，输入为单个站点存储的 IIS 日志所在的文件夹 (例如 `C:\inetpub\logs\LogFiles\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`，请键入您要使用的时间戳格式。有关支持的值的列表，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。有关支持的值的列表，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。有关支持的值的列表，请参阅 MSDN 上产品行为主题中表格中的 `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。如果此参数保留空白且时间戳不包括时区信息，则 CloudWatch Logs 默认为本地时区。如果时间戳已包含时区信息，则忽略此参数。
8. (可选) 对于 `LineCount`，在标头中键入行数以识别日志文件。例如，IIS 日志文件拥有几乎相同的标头。您可以输入 **5**，系统会读取日志文件标头的前五行以进行识别。在 IIS 日志文件中，第三行为日期和时间戳，但无法始终保证时间戳在日志文件之间是不同的。为此，建议包含至少一行实际日志数据以便对日志文件进行唯一指纹识别。

步骤 4：配置流程控制

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

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

请注意，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，并在后面的步骤中将其内容粘贴到另一个窗口中。

为 Systems Manager 创建 IAM 用户和角色

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。您可以选择性地创建用于配置和运行 Systems Manager 的唯一 IAM 用户账户。有关更多信息，请参阅 AWS Systems Manager 用户指南 中的为 Systems Manager 配置安全角色。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（适用于 Windows 实例） 中的将 IAM 角色附加到实例。

验证 Systems Manager 先决条件

在您使用 Systems Manager Run Command 配置与 CloudWatch Logs 的集成之前，请确保您的实例满足最低要求。有关更多信息，请参阅 AWS Systems Manager 用户指南 中的 Systems Manager 先决条件。

验证 Internet 访问权限

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

使用 Systems Manager Run Command 启用 CloudWatch Logs

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

使用 Run Command 配置与 CloudWatch Logs 的集成

1. 打开 Amazon EC2 控制台 https://console.amazonaws.cn/ec2/。
2. 在导航窗格中，选择 Systems Manager Services、Run Command。
3. 选择 Run a command。
4. 对于 Command document，选择 AWS-ConfigureCloudWatch。
5. 对于 Target instances，选择要与 CloudWatch Logs 集成的实例。如果您在此列表中未看到实例，则可能未针对 Run Command 配置实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Windows 实例）中的 Systems Manager 先决条件。
6. 对于 Status，选择 Enabled。
7. 对于 Properties，请复制并粘贴您在之前任务中创建的 JSON 内容。
8. 填写剩余选填字段并选择 Run。

使用以下过程查看 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 选项卡。
3. 选择 View Output。命令输出页面将显示命令执行的结果。

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

可以执行以下步骤，以允许运行 Windows Server 2012 和 Windows Server 2008 的实例将日志发送到 CloudWatch Logs。

下载示例配置文件

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

为 CloudWatch 配置 JSON 文件

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

步骤

步骤 1：启用 CloudWatch Logs

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

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

步骤 2：为 CloudWatch 配置设置

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

配置设置将日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 `CloudWatchLogs` 部分。

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

2. 将 `AccessKey` 和 `SecretKey` 字段留空。您将使用 IAM 角色配置凭证。
3. 对于 `Region`，键入要将日志数据发送到的区域 (例如 `us-east-2`)。
4. 对于 `LogGroup`，键入您的日志组的名称。此名称会显示在 CloudWatch 控制台的 Log Groups 屏幕上。
5. 对于 `LogStream`，键入目标日志流。此名称会显示在 CloudWatch 控制台的 Log Groups > Streams 屏幕上。

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

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

步骤 3：配置要发送的数据

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

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

1. 在 JSON 文件中，找到 `ApplicationEventLog` 部分。

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

2. 对于 `Levels`，指定要上传的消息类型。可以指定以下值之一：

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

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

将安全日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 `SecurityEventLog` 部分。

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

2. 对于 `Levels`，键入 **7** 以上传所有消息。

将系统事件日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 `SystemEventLog` 部分。

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

2. 对于 `Levels`，指定要上传的消息类型。可以指定以下值之一：

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

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

将其他类型的事件日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，添加新部分。每个部分必须有一个独立的 `Id`。

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

2. 对于 `Id`，键入要上传的日志的名称 (例如 **`WindowsBackup`**)。
3. 对于 `LogName`，请键入要上传的日志的名称。您可以按如下步骤找到日志的名称。
   a. 打开事件查看器。
   b. 在导航窗格中，选择 Applications and Services Logs。
   c. 导航到该日志，然后选择 Actions、Properties。
4. 对于 `Levels`，指定要上传的消息类型。可以指定以下值之一：

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

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

将 Windows 事件跟踪数据发送到 CloudWatch Logs

ETW (Windows 事件跟踪) 提供了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`，请键入您要使用的时间戳格式。有关支持的值的列表，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。有关支持的值的列表，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。有关支持的值的列表，请参阅 MSDN 上产品行为主题中表格中的 `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。如果此参数保留空白且时间戳不包括时区信息，则 CloudWatch Logs 默认为本地时区。如果时间戳已包含时区信息，则忽略此参数。
8. (可选) 对于 `LineCount`，在标头中键入行数以识别日志文件。例如，IIS 日志文件拥有几乎相同的标头。您可以输入 **5**，系统会读取日志文件标头的前三行以进行识别。在 IIS 日志文件中，第三行为日期和时间戳，但无法始终保证时间戳在日志文件之间是不同的。为此，建议包含至少一行实际日志数据以便对日志文件进行唯一指纹识别。

将 IIS 日志数据发送到 CloudWatch Logs

1. 在 JSON 文件中，找到 `IISLog` 部分。

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

2. 对于 `LogDirectoryPath`，输入为单个站点存储的 IIS 日志所在的文件夹 (例如 `C:\inetpub\logs\LogFiles\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`，请键入您要使用的时间戳格式。有关支持的值的列表，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。有关支持的值的列表，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。有关支持的值的列表，请参阅 MSDN 上产品行为主题中表格中的 `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。如果此参数保留空白且时间戳不包括时区信息，则 CloudWatch Logs 默认为本地时区。如果时间戳已包含时区信息，则忽略此参数。
8. (可选) 对于 `LineCount`，在标头中键入行数以识别日志文件。例如，IIS 日志文件拥有几乎相同的标头。您可以输入 **5**，系统会读取日志文件标头的前五行以进行识别。在 IIS 日志文件中，第三行为日期和时间戳，但无法始终保证时间戳在日志文件之间是不同的。为此，建议包含至少一行实际日志数据以便对日志文件进行唯一指纹识别。

步骤 4：配置流程控制

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

Warning

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

请注意，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。

启动代理

要让运行 Windows Server 2012 或 Windows Server 2008 的 Amazon EC2 实例将日志发送到 CloudWatch Logs，请使用 EC2Config 服务 (`EC2Config.exe`)。您的实例应具有 EC2Config 4.0 或更高版本并且您可以使用此过程。如果您的实例具有早期版本的 EC2Config，请参阅 Amazon EC2 用户指南（适用于 Windows 实例）中的 使用 EC2Config 3.x 或更低版本配置 CloudWatch

使用 EC2Config 4.x 配置 CloudWatch

1. 检查您在此过程前面编辑的 AWS.EC2.Windows.CloudWatch.json 文件的编码。只支持不含 BOM 的 UTF-8 编码。然后将文件保存在 Windows Server 2008 - 2012 R2 实例上的以下文件夹内：`C:\Program Files\Amazon\SSM\Plugins\awsCloudWatch\`。
2. 使用 Windows 服务控制面板或者使用以下 PowerShell 命令，来启动或重新启动 SSM 代理 (`AmazonSSMAgent.exe`)：

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

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

报告 CloudWatch Logs 代理的状态

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

报告代理状态

1. 连接到您的 EC2 实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的连接到您的实例。

   如果在连接时出现问题，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的排除连接实例的故障。
2. 在命令提示窗口中，键入以下命令：

```
sudo service awslogs status
```

   如果您运行的是 Amazon Linux 2，请键入以下命令：

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

启动 CloudWatch Logs 代理

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

启动代理

1. 连接到您的 EC2 实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的连接到您的实例。

   如果在连接时出现问题，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的排除连接实例的故障。
2. 在命令提示窗口中，键入以下命令：

```
sudo service awslogs start
```

   如果您运行的是 Amazon Linux 2，请键入以下命令：

```
sudo service awslogsd start
```

停止 CloudWatch Logs 代理

使用以下步骤停止 EC2 实例上的 CloudWatch Logs 代理。

停止代理

1. 连接到您的 EC2 实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的连接到您的实例。

   如果在连接时出现问题，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的排除连接实例的故障。
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
```

在 CloudWatch Logs 中创建日志组

当您使用 Amazon CloudWatch Logs User Guide前面章节中的步骤在 Amazon EC2 实例上安装 CloudWatch Logs 代理时，日志组将作为该过程的一部分创建。您还可以直接在 CloudWatch 控制台中创建日志组。

创建日志组

1. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。
3. 选择 Actions、Create log group。
4. 键入日志组的名称，然后选择 Create log group。

查看发送到 CloudWatch Logs 的日志数据

您可以查看和滚动浏览 CloudWatch Logs 代理发送到 CloudWatch Logs 的每一个日志流的日志数据。您可以指定要查看的日志数据的时间范围。

查看日志数据

1. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。
3. 对于 Log Groups，选择日志组以查看日志流。
4. 对于 Log Streams，选择日志流名称以查看日志数据。
5. 要更改日志数据的显示方式，请执行下列操作之一：
   - 要展开所有日志事件，请在日志事件列表上方选择 Expand all。
   - 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器。有关更多信息，请参阅 搜索和筛选日志数据 (p. 35)。
   - 要查看指定日期和时间范围的日志数据，请在日志事件列表上方选择 custom。您可以选择 Absolute 以指定日期和时间范围，也可以选择 Relative 以选择预定义的分钟数、小时数、天数或周数。还可以在 UTC 和 Local timezone 之间切换。

使用 AWS KMS 对 CloudWatch Logs 中的日志数据加密

可以使用 AWS Key Management Service (AWS KMS) 客户主键 (CMK) 对 CloudWatch Logs 中的日志数据加密。在创建日志组时或者在日志组已存在的情况下，通过将 CMK 与日志组相关联，在日志组级别启用加密。

将 CMK 与日志组关联后，该日志组所有新接收的数据都将使用 CMK 加密。此数据在其保留期内始终以加密格式存储。CloudWatch Logs 可随时根据请求解密该数据。无论何时请求加密数据，CloudWatch Logs 都必须拥有对 CMK 的权限。

解除 CMK 与日志组的关联后，CloudWatch Logs 将停止对日志组新接收的数据加密。所有先前接收的数据仍保持加密状态。

限制

- 要将 CMK 与日志组关联并执行以下步骤，您必须拥有以下权限：`kms:CreateKey`、`kms:GetKeyPolicy` 和 `kms:PutKeyPolicy`。
- 将 CMK 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 Logs 对关联 CMK 的访问权限或删除关联的 CMK，将无法再检索 CloudWatch Logs 中的加密数据。
- 不能使用 CloudWatch 控制台将 CMK 与日志组关联。

步骤 1：创建 AWS KMS CMK

要创建 AWS KMS CMK，请使用以下 create-key 命令：

```
aws kms create-key
```

输出包含 CMK 的密钥 ID 和 Amazon 资源名称 (ARN)。下面是示例输出：

```
{
    "KeyMetadata": {
        "KeyId": "6f815f63-e628-448c-8251-e40cb0d29f59",
        "Description": "",
        "Enabled": true,
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1478910250.94,
        "Arn": "arn:aws:kms:us-west-2:123456789012:key/6f815f63-e628-448c-8251-
e40cb0d29f59",
        "AWSAccountId": "123456789012"
    }
}
```

步骤 2：设置 CMK 的权限

默认情况下，所有 AWS KMS CMK 均为私有；只有资源所有者可以使用它加密和解密数据。但是，资源所有者可以将 CMK 的访问权限授予其他用户和资源。在此步中，您将授予 CloudWatch 服务委托人权限以使用该 CMK。此服务委托人必须位于存储 CMK 的相同区域内。

首先，使用以下 `policy.json`get-key-policy 命令将您的 CMK 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` 文件并添加粗体语句，将 *region* 替换为将用于您的日志组的区域，并使用逗号将现有语句与新语句分隔开。

```
{
  "Version" : "2012-10-17",
  "Id" : "key-default-1",
  "Statement" : [ {
      "Sid" : "Enable IAM User Permissions",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::880185128111:root"
```

```
      },
      "Action" : "kms:*",
      "Resource" : "*"
    },
    {
      "Effect": "Allow",
      "Principal": { "Service": "logs.region.amazonaws.com" },
      "Action": [
        "kms:Encrypt*",
        "kms:Decrypt*",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:Describe*"
      ],
      "Resource": "*"
    }
  ]
}
```

最后，使用以下 put-key-policy 命令添加更新后的策略：

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

步骤 3：将日志组与 CMK 关联

可以在创建日志组时或之后将 CMK 与它关联。

在创建日志组时将 CMK 与它关联

按如下方式使用 create-log-group 命令：

```
aws logs create-log-group --log-group-name my-log-group --kms-key-id "key-arn"
```

将 CMK 与现有日志组关联

按如下方式使用 associate-kms-key 命令：

```
aws logs associate-kms-key --log-group-name my-log-group --kms-key-id "key-arn"
```

步骤 4：解除日志组与 CMK 的关联

要解除 CMK 与日志组的关联，请使用以下 disassociate-kms-key 命令：

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

更改 CloudWatch Logs 中的日志数据保留期

默认情况下，日志数据将无限期存储在 CloudWatch Logs 中。但是，您可以配置要在日志组中存储日志数据多长时间。存储时间超过当前保留期设置的所有数据都将自动删除。您可以随时更改每个日志组的日志保留期。

更改日志保留期设置

1. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。

2. 在导航窗格中，选择 Logs。
3. 查找要更新的日志组。
4. 在该日志组的 Expire Events After 列中，选择当前保留期设置，例如 Never Expire。
5. 在 Edit Retention 对话框中，为 Retention 选择日志保留期值，然后选择 Ok。

标记 Amazon CloudWatch Logs 中的日志组

您可以使用标签的形式，将自己的元数据分配到您在 Amazon CloudWatch Logs 中创建的日志组。标签是您为日志组定义的键值对。使用标签是管理 AWS 资源和组织数据 (包括账单数据) 的一种简单却强有力的方式。

内容

有关标签的基本知识

可使用 AWS CLI 或 CloudWatch Logs API 完成以下任务：

- 在创建日志组时向其添加标签
- 向现有日志组添加标签
- 列出日志组的标签
- 删除日志组的标签

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 成本进行分类和跟踪。当您将标签应用于 AWS 资源 (包括日志组) 时，您的 AWS 成本分配报告将包括按标签汇总的使用率和成本。您可以设置代表业务类别 (例如成本中心、应用程序名称或所有者) 的标签，以便整理多种服务的成本。有关更多信息，请参阅 AWS Billing and Cost Management 用户指南 中的为自定义账单报告使用成本分配标签。

标签限制

以下限制适用于标签。

基本限制

- 每个日志组的最大标签数为 50。
- 标签键和值区分大小写。
- 无法更改或编辑已删除日志组的标签。

标签键限制

- 每个标签键必须是唯一的。如果您添加的标签具有已使用的键，则您的新标签将覆盖现有键值对。
- 标签键不能以 `aws:` 开头，因为此前缀将预留以供 AWS 使用。AWS 将代表您创建以此前缀开头的标签，但您不能编辑或删除这些标签。
- 标签键的长度必须介于 1 和 128 个 Unicode 字符之间。
- 标签键必须包含以下字符：Unicode 字母、数字、空格和以下特殊字符：`_ . / = + - @`。

标签值限制

- 标签值的长度必须介于 0 和 255 个 Unicode 字符之间。
- 标签值可以为空。另外，它们必须包含以下字符：Unicode 字母、数字、空格和以下任意特殊字符：`_ . / = + - @`。

使用 AWS CLI 标记日志组

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-log-group

为指定的日志组添加或更新标签。

list-tags-log-group

列出指定日志组的标签。

untag-log-group

删除指定日志组的标签。

使用 CloudWatch Logs API 标记日志组

您可以使用 CloudWatch Logs API 添加、列出和删除标签。有关示例，请参阅以下文档：

CreateLogGroup

创建日志组。您可以选择在创建日志组时添加标签。

TagLogGroup

为指定的日志组添加或更新标签。

ListTagsLogGroup

列出指定日志组的标签。

UntagLogGroup

删除指定日志组的标签。

搜索和筛选日志数据

CloudWatch Logs 代理开始将日志数据发布到 Amazon CloudWatch 后，您可以通过创建一个或多个指标筛选器开始搜索和筛选日志数据。指标筛选器定义在日志数据发送到 CloudWatch Logs 时要在日志数据中查找的字词和模式。CloudWatch Logs 使用这些指标筛选器将日志数据转换为 CloudWatch 数值指标，供您生成图表或设置警报。

筛选器不会以回溯方式筛选数据。筛选器只会发布在创建后发生的事件的指标数据点。筛选结果返回前 50 行，如果筛选结果上的时间戳早于指标创建时间，则不会显示这些行。

内容

概念

每个指标筛选器都由以下关键元素组成：

筛选器模式

一种符号描述，说明 CloudWatch Logs 应如何解释每个日志事件中的数据。例如，日志条目可能包含时间戳、IP 地址、字符串等。您可以使用模式来指定要在日志文件中查找的内容。

指标名称

CloudWatch 指标的名称，受监控的日志信息应该发布到此指标。例如，您可以向名为 ErrorCount 的指标发布信息。

指标命名空间

新 CloudWatch 指标的目标命名空间。

指标值

每次发现匹配日志时发布到指标的数字值。例如，如果您要对特定字词 (如“Error”) 的出现次数进行计数，则每出现一次，该值都将增加“1”。如果要计算传输的字节数，您可以按照在日志事件中找到的实际字节数累加。

默认值

在未找到匹配日志的时间段中报告给指标筛选器的值。通过将此项设置为 0，您可以确保在每个时间段中都报告了数据，防止出现存在无数据时间段的“断点”指标。

筛选器和模式语法

您需使用指标筛选器在日志事件中搜索和匹配字词、短语或值。当指标筛选器在日志事件中找到其中一个字词、短语或值时，您可以累加 CloudWatch 指标的值。例如，您可以创建一个指标筛选器以在日志事件中搜索 ERROR 并对这个词的出现次数进行计数。

指标筛选器还将从空格分隔的日志事件中提取数字值，例如 Web 请求的延迟。在这些示例中，您可以按照从日志中提取的实际数字值来累加指标值。

此外，您也可以使用条件运算符和通配符创建精确匹配。在创建指标筛选器之前，您可以在 CloudWatch 控制台中测试您的搜索模式。以下各个部分更为详细地描述了指标筛选器语法。

匹配事件日志中的字词

要搜索日志事件中的字词，可将该字词用作指标筛选器模式。您可以在指标筛选器模式中指定多个字词，但所有字词都必须出现在日志事件中才算是匹配。指标筛选器区分大小写。

包括字母数字或下划线之外其他字符的指标筛选器字词必须放在双引号 ("") 内。

要排除某个字词，可在该字词前使用减号 (-)。

示例 1：单个字词

“ERROR”筛选器模式匹配包含此字词的日志事件消息，例如以下内容：

- [ERROR] A fatal exception has occurred
- Exiting with ERRORCODE: -1

示例 2：包括一个字词，同时排除一个字词

在上面的示例中，如果您将筛选器模式更改为“ERROR”-“Exiting”，将会排除日志事件消息“Exiting with ERRORCODE: -1”。

示例 3：多个字词

“ERROR Exception”筛选器模式匹配同时包含这两个字词的日志事件消息，例如以下内容：

- [ERROR] Caught IllegalArgumentException
- [ERROR] Unhandled Exception

“Failed to process the request”筛选器模式匹配包含所有这些字词的日志事件消息，例如以下内容：

- [WARN] Failed to process the request
- [ERROR] Unable to continue: Failed to process the request

匹配 JSON 日志事件中的字词

您可以从 JSON 日志事件中提取值。要从 JSON 日志事件中提取值，您需要创建基于字符串的指标筛选器。不支持包含科学表示法的字符串。JSON 日志事件数据中的项目必须与指标筛选器完全匹配。您可能要在 JSON 日志事件中创建指标筛选器以便指示以下情况：

- 特定事件发生。例如 eventName 是“UpdateTrail”。
- IP 处于已知子网外部。例如，sourceIPAddress 不在某个已知子网范围内。
- 满足两个或更多其他条件的组合。例如，eventName 是“UpdateTrail”，并且 recipientAccountId 是 123456789012。

使用指标筛选器从 JSON 日志事件中提取值

您可以使用指标筛选器从 JSON 日志事件中提取值。指标筛选器检查传入日志，并在筛选器找到日志数据中的匹配时修改数字值。创建指标筛选器时，您可以每次在日志中找到匹配文本时累加计数，或者可以从日志中提取数值并使用这些值来累加指标值。

使用指标筛选器匹配 JSON 字词

JSON 日志事件的指标筛选语法使用以下格式：

```
{ SELECTOR EQUALITY_OPERATOR STRING }
```

指标筛选器必须包含在大括号 { } 内，以指示这是 JSON 表达式。指标筛选器包含以下部分：

SELECTOR

指定要检查的 JSON 属性。属性选择器始终以美元符号 ($) 开头，这表示 JSON 的根。属性选择器是字母数字字符串，它还支持“-”和“_”字符。数组元素通过 [NUMBER] 语法表示，必须跟在属性之后。示例：$.eventId、$.users[0]、$.users[0].id、$.requestParameters.instanceId。

EQUALITY_OPERATOR

可以是 = 或 !=.

STRING

带或不带引号的字符串。您可以在搜索词中、之前或之后使用星号“*”通配符来匹配任何文本。例如，*Event 将与 PutEvent 和 GetEvent 匹配。Event* 将与 EventId 和 EventName 匹配。Ev*ent 只与实际字符串 Ev*ent 匹配。完全由字母数字字符组成的字符串无需引号。包含 unicode 和其他字符 (如“@”、“$”、“\”等) 的字符串必须包含在双引号内才有效。

JSON 指标筛选器示例

以下是一个 JSON 示例：

```
{
  "eventType": "UpdateTrail",
  "sourceIPAddress": "111.111.111.111",
  "arrayKey": [
        "value",
        "another value"
  ],
  "objectList": [
       {
         "name": "a",
         "id": 1
       },
       {
         "name": "b",
         "id": 2
       }
  ],
  "SomeObject": null,
  "ThisFlag": true
}
```

以下筛选器将匹配：

```
{ $.eventType = "UpdateTrail" }
```

针对事件类型是 UpdateTrail 的筛选器。

```
{ $.sourceIPAddress != 123.123.* }
```

针对处于子网 123.123 前缀之外的 IP 地址的筛选器。

```
{ $.arrayKey[0] = "value" }
```

针对 arrayKey 中的第一个条目是“value”的筛选器。如果 arrayKey 不是数组，则为 false。

```
{ $.objectList[1].id = 2 }
```

针对 objectList 中的第二个条目的 id 属性 = 2 的筛选器。如果 objectList 不是数组，则为 false。如果 objectList 中的项目不是对象或者没有 id 属性，则为 false。

```
{ $.SomeObject IS NULL }
```

针对 SomeObject 设置为 null 的筛选器。仅当指定的对象为 null 时，此条件才成立。

```
{ $.SomeOtherObject NOT EXISTS }
```

针对 SomeOtherObject 不存在的筛选器。仅当指定的对象在日志数据中不存在时，此条件才成立。

```
{ $.ThisFlag IS TRUE }
```

针对 ThisFlag 为 TRUE 的筛选器。这也适用于看检查是否有 FALSE 值的布尔筛选器。

JSON 复合条件

您可以使用 OR (||) 和 AND (&&) 将多个条件合并为复合表达式。允许使用圆括号，语法遵循标准运算顺序 () > && > ||。

```
{
    "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]
    ]
}
```

示例

```
{ ($.user.id = 1) && ($.users[0].email = "John.Doe@example.com") }
```

与以上 JSON 匹配。

```
{ ($.user.id = 2 && $.users[0].email = "nonmatch") || $.actions[2] = "GET" }
```

与以上 JSON 不匹配。

```
{ $.user.email = "John.Stiles@example.com" || $.coordinates[0][1] = nonmatch &&
 $.actions[2] = nomatch }
```

与以上 JSON 匹配。

```
{ ($.user.email = "John.Stiles@example.com" || $.coordinates[0][1] = nonmatch) &&
 $.actions[2] = nomatch }
```

与以上 JSON 不匹配。

JSON 特殊注意事项

SELECTOR 必须指向 JSON 中的一个值节点 (字符串或数字)。如果它指向数组或对象，则不会应用筛选器，因为日志格式与筛选器不匹配。例如，{$.users = 1} 和 {$.users != 1} 都无法与其中用户是数组的日志事件匹配：

```
{
  "users": [1, 2, 3]
}
```

数字比较

指标筛选器语法支持数字比较的精确匹配。支持以下数字比较：<、>、>=、<=、=、!=

数字筛选器的语法为

```
{ SELECTOR NUMERIC_OPERATOR NUMBER }
```

指标筛选器必须包含在大括号 { } 内，以指示这是 JSON 表达式。指标筛选器包含以下部分：

SELECTOR

指定要检查的 JSON 属性。属性选择器始终以美元符号 ($) 开头，这表示 JSON 的根。属性选择器是字母数字字符串，它还支持“-”和“_”字符。数组元素通过 [NUMBER] 语法表示，必须跟在属性之后。示例：$.latency、$.numbers[0]、$.errorCode、$.processes[4].averageRuntime。

NUMERIC_OPERATOR

可能是以下值之一：=、!=, <, >, <=, 或者 >=.

NUMBER

带可选 + 或 - 符号的整数、带可选 + 或 - 符号的小数或采用科学表示法的数字 (带可选 + 或 - 符号的整数或小数，后跟“e”，后跟带可选 + 或 - 符号的整数)。

示例：

```
{ $.latency >= 500 }
{ $.numbers[0] < 10e3 }
{ $.numbers[0] < 10e-3 }
```

```
{ $.processes[4].averageRuntime <= 55.5 }
{ $.errorCode = 400 }
{ $.errorCode != 500 }
{ $.latency > +1000 }
```

使用指标筛选器从空格分隔的日志事件中提取值

您可以使用指标筛选器从空格分隔的日志事件中提取值。将一对方括号 [] 或两个双引号 ("") 之间的字符视为单个字段。例如：

```
127.0.0.1 - frank [10/Oct/2000:13:25:15 -0700] "GET /apache_pb.gif HTTP/1.0" 200 1534
127.0.0.1 - frank [10/Oct/2000:13:35:22 -0700] "GET /apache_pb.gif HTTP/1.0" 500 5324
127.0.0.1 - frank [10/Oct/2000:13:50:35 -0700] "GET /apache_pb.gif HTTP/1.0" 200 4355
```

要指定解析空格分隔的事件的指标筛选器模式，指标筛选器模式必须使用名称指定字段，以逗号隔开，且整个模式包括在方括号中。例如：[ip, user, username, timestamp, request, status_code, bytes]。

如果您不知道字段数，可以使用省略号 (...) 进行快速输入。例如：

```
[..., status_code, bytes]
[ip, user, ..., status_code, bytes]
[ip, user, ...]
```

您还可以对您的字段添加条件，以使只有满足所有条件的日志事件才能与筛选器匹配。例如：

```
[ip, user, username, timestamp, request, status_code, bytes > 1000]
[ip, user, username, timestamp, request, status_code = 200, bytes]
[ip, user, username, timestamp, request, status_code = 4*, bytes]
[ip, user, username, timestamp, request = *html*, status_code = 4*, bytes]
```

CloudWatch Logs 支持字符串和数字条件字段。对于字符串字段，您可以将 = 或 != 运算符和星号 (*) 一起使用。

对于数字字段，您可以使用 >、<、>=、<=、= 和 != 运算符。

如果您使用的是以空格分隔的筛选条件，提取的域将映射到以空格分隔的域的名称 (如筛选条件中所示)，再映射到所有这些域的值。如果您使用的不是以空间分隔的筛选条件，这将为空。

示例筛选条件：

```
[..., request=*.html*, status_code=4*,]
```

筛选条件的示例日志事件：

```
127.0.0.1 - frank [10/Oct/2000:13:25:15 -0700] \"GET /index.html HTTP/1.0\" 404 1534
```

日志事件和筛选器模式的提取的域：

```
{
   "$status_code": "404",
   "$request": "GET /products/index.html HTTP/1.0",
   "$7": "1534",
   "$4": "10/Oct/2000:13:25:15 -0700",
   "$3": "frank",
   "$2": "-",
   "$1": "127.0.0.1"
```

```
}
```

设置在发现匹配项时如何更改指标值

指标筛选器在日志事件中找到一个匹配字词、短语或值时，它会按照您为“Metric Value”指定的数量累加 CloudWatch 指标中的计数。指标值每分钟汇总并报告。

如果在 1 分钟时间段内提取日志，但未找到任何匹配项，则报告为 Default Value 指定的值 (如果有)。不过，如果在 1 分钟时间段内未提取日志事件，则不会报告任何值。

指定 Default Value (即使该值为 0) 可帮助确保更频繁地报告数据，从而在未找到匹配项时帮助防止断点指标。

例如，假设有一个日志组，每分钟发布两条记录，并且“Metric Value”为 1，“Default Value”为 0。如果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。如果第二分钟内发布的日志记录中没有匹配项，则为两个日志记录均使用“Default Value”(即 0)，该分钟的指标值为 0。

如果您未指定“Default Value”，则对于未找到模式匹配的任意时间段，不报告任何数据。

发布日志条目中找到的数值

您也可以使用指标筛选器根据在日志中找到的数值来发布值，而不是计算在日志中找到的匹配项的数目。以下步骤介绍了如何使用在 JSON 请求 `metricFilter: { $.latency = * } metricValue: $.latency` 中发现的延迟发布指标。

使用 JSON 请求中的延迟发布指标

1. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。
3. 在内容窗格中，选择一个日志组，然后选择 Create Metric Filter。
4. 在 Define Logs Metric Filter 屏幕上，为 Filter Pattern 键入 **`{ $.latency = * }`**，然后选择 Assign Metric。
5. 在 Create Metric Filter and Assign a Metric 屏幕上，选择 Show advanced metric settings。
6. 为 Metric Name 键入 myMetric。
7. 对于 Metric Value，输入 **`$.latency`**。
8. 对于 Default Value，键入 0，然后选择 Create Filter。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。

以下日志事件会在过滤器创建之后将值 50 发布到指标 myMetric。

```
{
"latency": 50,
"requestType": "GET"
}
```

创建指标筛选器

以下示例介绍如何创建指标筛选器。

示例

- 示例：对日志事件进行计数 (p. 42)

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，名为 MyAppAccessCount 的指标筛选器应用于日志组 MyApp/access.log，以在 CloudWatch 命名空间 MyNamespace 中创建指标 EventCount。该筛选器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. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。
3. 在内容窗格中，选择一个日志组，然后选择 Create Metric Filter。
4. 在 Define Logs Metric Filter 屏幕上，将 Filter Pattern 留空。
5. 选择 Assign Metric，然后在 Create Metric Filter and Assign a Metric 屏幕上，为 Filter Name 键入 **`EventCount`**。
6. 在 Metric Details 下，为 Metric Namespace 键入 **`MyNameSpace`**。
7. 对于 Metric Name，键入 **`MyAppEventCount`**。
8. 选择 Show advanced metric settings 并确认 Metric Value 为 1。这指定对于每个日志事件，计数以 1 累加。
9. 对于 Default Value，键入 0，然后选择 Create Filter。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。

使用 AWS CLI 创建指标筛选器

在命令提示符处，运行以下命令：

```
aws logs put-metric-filter \
  --log-group-name MyApp/access.log \
  --filter-name EventCount \
  --filter-pattern "" \
  --metric-transformations \
  metricName=MyAppEventCount,metricNamespace=MyNamespace,metricValue=1,defaultValue=0
```

您可以通过发布任何事件数据来测试此新策略。您应该看到有数据点发布到指标 MyAppAccessEventCount。

使用 AWS CLI 发布事件数据

在命令提示符处，运行以下命令：

```
aws logs put-log-events \
  --log-group-name MyApp/access.log --log-stream-name TestStream1 \
  --log-events \
    timestamp=1394793518000,message="Test event 1" \
    timestamp=1394793518000,message="Test event 2" \
    timestamp=1394793528000,message="This message also contains an Error"
```

示例：对字词的出现次数进行计数

日志事件经常包含您需要计数的重要消息，可能是关于操作的成功或失败的消息。例如，如果给定操作失败，可能发生错误，且错误记录到日志文件中。您可能需要监控这些日志条目以了解错误发生趋势。

在以下示例中，创建了一个指标筛选器来监控“Error”这个词。策略已创建并添加到日志组 MyApp/message.log。对于每个包含“Error”的事件，CloudWatch Logs 都将数据点发布到 MyApp/message.log 命名空间中值为“1”的 CloudWatch 自定义指标 ErrorCount。如果没有任何事件包含“Error”这个单词，则发布值 0。当在 CloudWatch 控制台中绘制该数据的图表时，请务必使用总计统计数据。

使用 CloudWatch 控制台创建指标筛选器

1. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。
3. 在内容窗格中，选择一个日志组，然后选择 Create Metric Filter。
4. 在 Define Logs Metric Filter 屏幕上，为 Filter Pattern 键入 **`Error`**。

   Note

   Filter Pattern 中的所有条目都区分大小写。
5. 要测试筛选器模式，请为 Select Log Data to Test 选择要对其测试指标筛选器的日志组，然后选择 Test Pattern。
6. 在 Results 下，CloudWatch Logs 显示一条消息，表明筛选器模式在日志文件中出现的次数。

   要查看详细结果，请选择 Show test results。
7. 选择 Assign Metric，然后在 Create Metric Filter and Assign a Metric 屏幕上，为 Filter Name 键入 **`MyAppErrorCount`**。
8. 在 Metric Details 下，为 Metric Namespace 键入 MyNameSpace。
9. 为 Metric Name 键入 ErrorCount。
10. 选择 Show advanced metric settings 并确认 Metric Value 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，键入 0，然后选择 Create Filter。

使用 AWS CLI 创建指标筛选器

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=EventCount,metricNamespace=MyNamespace,metricValue=1,defaultValue=0
```

您可以通过在消息中发布包含“Error”这个词的事件来测试此新策略。

使用 AWS CLI 发布事件

在命令提示符处，运行以下命令。注意，模式区分大小写。

```
aws logs put-log-events \
  --log-group-name MyApp/access.log --log-stream-name TestStream1 \
  --log-events \
    timestamp=1394793518000,message="This message contains an Error" \
    timestamp=1394793528000,message="This message also contains an Error"
```

示例：对 HTTP 404 代码进行计数

使用 CloudWatch Logs，您可以监控 Apache 服务器返回 HTTP 404 响应的次数，404 是表示找不到页面的响应代码。您可能需要对此进行监控，从而了解站点来访者找不到所查找资源的频率。假定日志记录是结构化的，每一个日志记录 (站点访问) 都包含以下信息：

- 请求者 IP 地址
- RFC 1413 标识
- Username
- 时间戳
- 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. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。
3. 在内容窗格中，选择一个日志组，然后选择 Create Metric Filter。
4. 在 Define Logs Metric Filter 屏幕上，为 Filter Pattern 键入 **`[IP, UserInfo, User, Timestamp, RequestInfo, StatusCode=404, Bytes]`**。
5. 要测试筛选器模式，请为 Select Log Data to Test 选择要对其测试指标筛选器的日志组，然后选择 Test Pattern。
6. 在 Results 下，CloudWatch Logs 显示一条消息，表明筛选器模式在日志文件中出现的次数。

   要查看详细结果，请选择 Show test results。
7. 选择 Assign Metric，然后在 Create Metric Filter and Assign a Metric 屏幕上，为 Filter Name 键入 HTTP404Errors。
8. 在 Metric Details 下，为 Metric Namespace 键入 **`MyNameSpace`**。
9. 对于 Metric Name，键入 **`ApacheNotFoundErrorCount`**。
10. 选择 Show advanced metric settings 并确认 Metric Value 为 1。这指定对于每个“404 Error”事件，计数以 1 累加。
11. 对于 Default Value，键入 0，然后选择 Create Filter。

使用 AWS CLI 创建指标筛选器

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器的创建。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"
```

在发布这些日志事件示例后，您很快可以在 CloudWatch 控制台中检索 ApacheNotFoundErrorCount 指标。

示例：对 HTTP 4xx 代码进行计数

如前面的示例一样，您可能需要监控 Web 服务访问日志和监控 HTTP 响应代码级别。例如，您可能需要监控所有 HTTP 400 级的错误。但是，您可能不想为每一个返回代码指定新的指标筛选器。

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 (p. 44) 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. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。
3. 在内容窗格中，选择一个日志组，然后选择 Create Metric Filter。
4. 在 Define Logs Metric Filter 屏幕上，为 Filter Pattern 键入 **`[ip, id, user, timestamp, request, status_code=4*, size]`**。
5. 要测试筛选器模式，请为 Select Log Data to Test 选择要对其测试指标筛选器的日志组，然后选择 Test Pattern。
6. 在 Results 下，CloudWatch Logs 显示一条消息，表明筛选器模式在日志文件中出现的次数。

   要查看详细结果，请单击 Show test results。
7. 选择 Assign Metric，然后在 Create Metric Filter and Assign a Metric 屏幕上，为 Filter Name 键入 **`HTTP4xxErrors`**。

8. 在 Metric Details 下，为 Metric Namespace 键入 **MyNameSpace**。
9. 对于 Metric Name，键入 HTTP4xxErrors。
10. 选择 Show advanced metric settings 并确认 Metric Value 为 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，键入 0，然后选择 Create Filter。

使用 AWS CLI 创建指标筛选器

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 日志提取字段

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。

此提取规则与日志事件的 7 个字段匹配。指标值是第七个匹配的标记的值。您可以在提取规则的 `metricValue` 字段中看到对该标记的引用为“$7”。

使用 CloudWatch 控制台创建指标筛选器

1. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。
3. 在内容窗格中，选择一个日志组，然后选择 Create Metric Filter。
4. 在 Define Logs Metric Filter 屏幕上，为 Filter Pattern 键入 **[ip, id, user, timestamp, request, status_code, size]**。
5. 要测试筛选器模式，请为 Select Log Data to Test 选择要对其测试指标筛选器的日志组，然后选择 Test Pattern。
6. 在 Results 下，CloudWatch Logs 显示一条消息，表明筛选器模式在日志文件中出现的次数。

   要查看详细结果，请单击 Show test results。
7. 选择 Assign Metric，然后在 Create Metric Filter and Assign a Metric 屏幕上，为 Filter Name 键入 size。
8. 在 Metric Details 下，为 Metric Namespace 键入 **MyNameSpace**。
9. 为 Metric Name 键入 **BytesTransferred**
10. 选择 Show advanced metric settings，为 Metric Value 键入 **$size**。
11. 对于 Default Value，键入 0，然后选择 Create Filter。

使用 AWS CLI 创建指标筛选器

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=4*, size]' \
--metric-transformations \
metricName=BytesTransferred,metricNamespace=MyNamespace,metricValue=$size,defaultValue=0
```

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
```

列出指标筛选器

您可以列出日志组的所有指标筛选器。

使用 CloudWatch 控制台列出指标筛选器

1. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。
3. 在内容窗格的日志组列表中，在 Metric Filters 列中选择筛选器数目。

   Log Groups > Filters for 屏幕将列出与该日志组关联的所有指标筛选器。

使用 AWS CLI 列出指标筛选器

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]"
        }
    ]
```

```
}
```

删除指标筛选器

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. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。
3. 在内容窗格的 Metric Filter 列中，选择指标筛选器。
4. 在 Logs Metric Filters 屏幕上的指标筛选器中，选择 Delete Filter。
5. 当系统提示进行确认时，选择 Yes, Delete。

使用 AWS CLI 删除指标筛选器

在命令提示符处，运行以下命令：

```
aws logs delete-metric-filter --log-group-name MyApp/access.log \
 --filter-name MyFilterName
```

使用筛选器模式搜索日志数据

可以使用 筛选器和模式语法 (p. 35) 搜索日志数据。您可以搜索日志组中的所有日志流，也可以使用 AWS CLI 搜索特定日志流。每次搜索运行时，都会返回到所找到数据的第一页，并且使用令牌来检索下一页数据或继续搜索。如果没有返回任何结果，可以继续搜索。

您可以设置要查询的时间范围来限制搜索范围。您可以从较大范围开始，看看感兴趣的日志行在何处，然后缩短时间范围在相应时间范围内查看您感兴趣的日志。

您还可以通过从日志提取的指标直接定向至相应日志。

使用控制台搜索日志条目

您可以使用控制台搜索满足指定条件的日志条目。

使用控制台搜索日志

1. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。
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. 对于 Filter，键入要使用的指标筛选器语法，然后按 Enter。

使用控制台搜索某个时间范围的所有日志条目

1. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。

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 Events。
5. 对于 Filter，键入要使用的指标筛选器语法，选择日期和时间范围，然后按 Enter。

使用 AWS CLI 搜索日志条目

您可以使用 AWS CLI 搜索满足指定条件的日志条目。

使用 AWS CLI 搜索日志条目

在命令提示符处，运行以下 filter-log-events 命令。使用 `--filter-pattern` 将结果限制为指定的筛选器模式，并使用 `--log-stream-names` 将结果限制为指定的日志组。

```
aws logs filter-log-events --log-group-name my-group [--log-stream-
names LIST_OF_STREAMS_TO_SEARCH] --filter-pattern VALID_METRIC_FILTER_PATTERN]
```

使用 AWS CLI 搜索给定时间范围内的日志条目

在命令提示符处，运行以下 filter-log-events 命令：

```
aws logs filter-log-events --log-group-name my-group [--log-stream-
names LIST_OF_STREAMS_TO_SEARCH] [--start-time 1482197400000] [--end-time 1482217558365]
 [--filter-pattern VALID_METRIC_FILTER_PATTERN]
```

从指标定向至日志

您可以从控制台的其他部分转到特定的日志条目。

从控制面板小部件转到日志

1. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Dashboards。
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器，请从列表中选择一个。如果有更多指标筛选器，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器。

从指标转到日志

1. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Metrics。
3. 在 All metrics 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、View logs。如果存在多个指标筛选器，请从列表中选择一个。如果有更多指标筛选器，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器。

故障排除

搜索耗时太长

如果有大量日志数据，搜索可能需要很长时间才能完成。要提高搜索速度，可以执行以下操作：

- 如果使用 AWS CLI，则可以将搜索限制为仅搜索您感兴趣的日志流。例如，如果日志组有 1000 个日志流，但您只需要查看三个已知相关的日志流，则可以使用 AWS CLI 将搜索限制为仅搜索相应日志组中的那三个日志流。
- 使用更短、更细粒度的时间范围，从而减少搜索的数据量和加快查询速度。

使用订阅实时处理日志数据

您可以使用订阅从 CloudWatch Logs 中访问日志事件的实时源并将其传输到其他服务 (如 Amazon Kinesis 流、Amazon Kinesis Data Firehose 流或 AWS Lambda)，以进行自定义处理、分析或加载到其他系统中。要开始订阅日志事件，请创建用于接收事件的接收源，例如 Kinesis 流。订阅筛选器定义了筛选器模式 (用于筛选传输到您的 AWS 资源的日志事件)，以及有关要将匹配的日志事件发送到的位置的信息。

CloudWatch Logs 还会生成有关将日志事件转发到订阅的 CloudWatch 指标。有关更多信息，请参阅 Amazon CloudWatch Logs 指标与维度。

内容

概念

每个订阅筛选器都由以下关键元素组成：

日志组名称

要将订阅筛选器关联到的日志组。如果筛选器模式与日志事件匹配，要上传到此日志组的所有事件日志都受订阅筛选器的约束并将传输到所选的 Kinesis 流。

筛选器模式

一种符号描述，说明 CloudWatch Logs 应如何解释每个日志事件中的数据以及可限制传输到目标 AWS 资源的内容的筛选表达式。有关筛选器模式语法的更多信息，请参阅筛选器和模式语法 (p. 35)。

目标 ARN

要用作订阅源目标的 Kinesis 流、Kinesis Data Firehose 流或 Lambda 函数的 Amazon 资源名称 (ARN)。

角色 ARN

一个 IAM 角色，用于向 CloudWatch Logs 授予将数据放入所选 Kinesis 流的必要权限。此角色不适用于 Lambda 目标，因为 CloudWatch Logs 可以通过 Lambda 函数本身的访问控制设置获得必要的权限。

分配

当目标是 Amazon Kinesis 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。

使用 CloudWatch Logs 订阅筛选器

可以对 Kinesis、Lambda 或 Kinesis Data Firehose 使用订阅筛选器。

示例

示例 1：Kinesis 订阅筛选器

以下示例将订阅筛选器与包含 AWS CloudTrail 事件的日志组关联，以便将记录中由“根”AWS 凭证完成的每个活动传输到名为“RootAccess”的 Kinesis 流。有关如何将 AWS CloudTrail 事件发送到 CloudWatch Logs 的更多信息，请参阅 AWS CloudTrail User Guide 中的将 CloudTrail 事件发送到 CloudWatch Logs。

为 Kinesis 创建订阅筛选器

1. 使用以下命令创建目标 Kinesis 流：

```
$ C:\>  aws kinesis create-stream --stream-name "RootAccess" --shard-count 1
```

2. 请耐心等待，直到 Kinesis 流变为活动状态 (这可能需要一两分钟)。您可使用以下 Kinesis describe-stream 命令检查 StreamDescription.StreamStatus 属性。此外，请记下 StreamDescription.StreamARN 值，因为后面的步骤中将会用到它：

```
aws kinesis describe-stream --stream-name "RootAccess"
```

   下面是示例输出：

```
{
    "StreamDescription": {
        "StreamStatus": "ACTIVE",
        "StreamName": "RootAccess",
        "StreamARN": "arn:aws:kinesis:us-east-1:123456789012:stream/RootAccess",
        "Shards": [
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "EndingHashKey": "340282366920938463463374607431768211455",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
                    "49551135218688818456679503831981458784591352702181572610"
                }
            }
        ]
    }
}
```

3. 创建 IAM 角色，该角色将向 CloudWatch Logs 授予将数据放入 Kinesis 流的权限。首先，您需要在文件 (例如 `~/TrustPolicyForCWL.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。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

4. 使用 create-role 命令创建 IAM 角色，并指定信任策略文件。请记下返回的 Role.Arn 值，因为后面的步骤中将会用到它：

```
aws iam create-role --role-name CWLtoKinesisRole --assume-role-policy-document file://
~/TrustPolicyForCWL.json
```

```
{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.us-east-1.amazonaws.com"
                }
            }
        },
        "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可对您的账户执行的 CloudWatch Logs 操作。首先，您将在文件 (例如 `~/PermissionsForCWL.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:us-east-1:123456789012:stream/RootAccess"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/CWLtoKinesisRole"
    }
  ]
}
```

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.json
```

7. 在 Kinesis 流进入 Active 状态并且您已创建 IAM 角色后，您便可以创建 CloudWatch Logs 订阅筛选器。订阅筛选器将立即让实时日志数据开始从所选日志组流动到您的 Kinesis 流：

```
aws logs put-subscription-filter \
    --log-group-name "CloudTrail" \
    --filter-name "RootAccess" \
    --filter-pattern "{$.userIdentity.type = Root}" \
    --destination-arn "arn:aws:kinesis:us-east-1:123456789012:stream/RootAccess" \
    --role-arn "arn:aws:iam::123456789012:role/CWLtoKinesisRole"
```

8. 设置订阅筛选器后，CloudWatch Logs 会将与筛选器模式匹配的所有传入事件日志转发到您的 Kinesis 流。您可以通过抓取 Kinesis 分片迭代器并使用 Kinesis get-records 命令获取一些 Kinesis 记录来验证此事件是否发生：

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

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

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

请注意，您可能需要在 Kinesis 开始返回数据之前调用几次此命令。

您将看到包含一组记录的响应。Kinesis 记录中的 Data 属性使用 gzip 格式进行了 Base64 编码和压缩。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",
    "logStream": "111111111111_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器名称的列表。

messageType

数据消息将使用“DATA_MESSAGE”类型。有时候，CloudWatch Logs 可能发出类型为“CONTROL_MESSAGE”的 Kinesis 记录，这主要是为了检查目标是否可访问。

logEvents

表示为一组日志事件记录的实际日志数据。“id”属性是每个日志事件的唯一标识符。

示例 2：AWS Lambda 订阅筛选器

此示例将创建一个 CloudWatch Logs 订阅筛选器，用于将日志数据发送到您的 AWS Lambda 函数。

为 Lambda 创建订阅筛选器

1. 创建 AWS Lambda 函数。

   确保您已设置 Lambda 执行角色。有关详细信息，请参阅以下指南中的 步骤 2.2：创建 IAM 角色 (执行角色)：AWS Lambda Developer Guide。
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 = new Buffer(input.awslogs.data, 'base64');
    zlib.gunzip(payload, function(e, result) {
        if (e) {
            context.fail(e);
        } else {
            result = JSON.parse(result.toString('ascii'));
            console.log("Event Data:", JSON.stringify(result, null, 2));
            context.succeed();
        }
    });
};
```

3. 压缩 helloWorld.js 文件，并用名称 `helloWorld.zip` 保存它。
4. 使用以下命令，其中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 file://file-path/helloWorld.zip \
    --role lambda-execution-role-arn \
    --handler helloWorld.handler \
    --runtime nodejs4.3
```

5. 授予 CloudWatch Logs 执行您的函数的权限。使用以下命令，将占位符账户替换为您自己的账户，将占位符日志组替换为要处理的日志组：

```
aws lambda add-permission \
```

```
    --function-name "helloworld" \
    --statement-id "helloworld" \
    --principal "logs.us-east-1.amazonaws.com" \
    --action "lambda:InvokeFunction" \
    --source-arn "arn:aws:logs:us-east-1:123456789123:log-group:TestLambda:*" \
    --source-account "123456789012"
```

6. 使用以下命令创建订阅筛选器，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:us-east-1:123456789123: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，请导航至 Lambda 函数，在此处您可以查看 /aws/lambda/helloworld 中的输出：

```
aws logs put-log-events --log-group-name myLogGroup --log-stream-name stream1 --log-
events "[{\"timestamp\":<CURRENT TIMESTAMP MILLIS> , \"message\": \"Simple Lambda
 Test\"}]"
```

   您将看到包含一组 Lambda 的响应。Lambda 记录中的 Data 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器名称的列表。

messageType

数据消息将使用“DATA_MESSAGE”类型。有时候，CloudWatch Logs 可能发出类型为“CONTROL_MESSAGE”的 Lambda 记录，这主要是为了检查目标是否可访问。

logEvents

表示为一组日志事件记录的实际日志数据。“id”属性是每个日志事件的唯一标识符。

示例 3：Amazon Kinesis Data Firehose 订阅筛选器

此示例将创建一个 CloudWatch Logs 订阅，用于将与所定义筛选器匹配的所有传入日志事件发送到您的 Amazon Kinesis Data Firehose 传输流。已经使用 gzip 6 级压缩对从 CloudWatch Logs 发送到 Amazon Kinesis Data Firehose 的数据进行压缩，因此您无需在 Kinesis Data Firehose 传输流中使用压缩。

为 Kinesis Data Firehose 创建订阅筛选器

1. 创建一个 Amazon Simple Storage Service (Amazon S3) 存储桶。我们建议您使用专为 CloudWatch Logs 创建的存储桶。 但是，如果要使用现有存储桶，请跳至第 2 步。

   运行以下命令，将占用符区域替换为您想使用的区域：

   ```
   aws s3api create-bucket --bucket my-bucket --create-bucket-configuration
    LocationConstraint=us-east-1
   ```

   下面是示例输出：

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

2. 创建 IAM 角色，该角色将向 Amazon Kinesis Data Firehose 授予将数据放入您的 Amazon S3 存储桶的权限。

   有关更多信息，请参阅 Amazon Kinesis Data Firehose 开发人员指南 中的使用 Amazon Kinesis Data Firehose 控制访问权限。

   首先，如下所示使用文本编辑器在文件 `~/TrustPolicyForFirehose.json` 中创建信任策略，将 account-id 替换为您的 AWS 账户 ID：

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "firehose.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": { "StringEquals": { "sts:ExternalId":"account-id" } }
  }
  }
```

3. 使用 create-role 命令创建 IAM 角色，并指定信任策略文件。请记下返回的 Role.Arn 值，因为后面的步骤中将会用到它：

```
aws iam create-role \
      --role-name FirehosetoS3Role \
      --assume-role-policy-document file://~/TrustPolicyForFirehose.json

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.us-east-1.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可对您的账户执行的 Kinesis Data Firehose 操作。首先，使用文本编辑器在文件 ~/PermissionsForFirehose.json 中创建权限策略：

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "s3:AbortMultipartUpload",
          "s3:GetBucketLocation",
          "s3:GetObject",
          "s3:ListBucket",
          "s3:ListBucketMultipartUploads",
          "s3:PutObject" ],
      "Resource": [
          "arn:aws:s3:::my-bucket",
          "arn:aws:s3:::my-bucket/*" ]
    }
  ]
}
```

5. 使用以下 put-role-policy 命令将权限策略与角色关联：

```
aws iam put-role-policy --role-name FirehosetoS3Role --policy-name Permissions-Policy-For-Firehose --policy-document file://~/PermissionsForFirehose.json
```

6. 如下所示创建目标 Kinesis Data Firehose 传输流，将 RoleARN 和 BucketARN 的占位符值替换为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:::my-bucket'
```

请注意，Kinesis Data Firehose 对所传输的 Amazon S3 对象自动使用 YYYY/MM/DD/HH UTC 时间格式的前缀。您可以指定要添加在该时间格式前缀前面的额外前缀。如果前缀以正斜杠 (/) 结束，则在 Amazon S3 存储桶中显示为文件夹。

7. 请耐心等待，直到流变为活动状态 (这可能需要几分钟时间)。您可以使用 Kinesis Data Firehose describe-delivery-stream 命令检查 DeliveryStreamDescription.DeliveryStreamStatus 属性。此外，请记下 DeliveryStreamDescription.DeliveryStreamARN 值，因为后面的步骤中将会用到它：

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
{
    "DeliveryStreamDescription": {
        "HasMoreDestinations": false,
        "VersionId": "1",
        "CreateTimestamp": 1446075815.822,
        "DeliveryStreamARN": "arn:aws:firehose:us-east-1:123456789012:deliverystream/
my-delivery-stream",
        "DeliveryStreamStatus": "ACTIVE",
        "DeliveryStreamName": "my-delivery-stream",
        "Destinations": [
            {
                "DestinationId": "destinationId-000000000001",
                "S3DestinationDescription": {
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "RoleARN": "delivery-stream-role",
                    "BucketARN": "arn:aws:s3:::my-bucket",
                    "BufferingHints": {
                        "IntervalInSeconds": 300,
                        "SizeInMBs": 5
                    }
                }
            }
        ]
    }
    }
```

8. 创建 IAM 角色，该角色将向 CloudWatch Logs 授予将数据放入 Kinesis Data 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：

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
  }
```

9. 使用 create-role 命令创建 IAM 角色，并指定信任策略文件。请记下返回的 Role.Arn 值，因为后面的步骤中将会用到它：

```
aws iam create-role \
      --role-name CWLtoKinesisFirehoseRole \
      --assume-role-policy-document file://~/TrustPolicyForCWL.json

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.us-east-1.amazonaws.com"
                }
            }
        },
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2015-05-29T13:46:29.431Z",
        "RoleName": "CWLtoKinesisFirehoseRole",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
    }
}
```

10. 创建权限策略以定义可对您的账户执行的 CloudWatch Logs 操作。首先，使用文本编辑器创建权限策略文件 (例如 `~/PermissionsForCWL.json`)：

```
{
    "Statement":[
      {
        "Effect":"Allow",
        "Action":["firehose:*"],
        "Resource":["arn:aws:firehose:us-east-1:123456789012:*"]
      },
      {
        "Effect":"Allow",
        "Action":["iam:PassRole"],
        "Resource":["arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"]
      }
    ]
}
```

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 Kinesis Data Firehose 传输流进入活动状态并且您已创建 IAM 角色后，您便可以创建 CloudWatch Logs 订阅筛选器。订阅筛选器将立即启动从所选日志组到您的 Amazon Kinesis 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:us-east-1:123456789012:deliverystream/my-delivery-stream" \
    --role-arn "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
```

13. 当您设置订阅筛选器后，CloudWatch Logs 将与筛选器模式匹配的所有传入事件日志转发到您的 Amazon Kinesis Data Firehose 传输流。您的数据将开始根据您的 Amazon Kinesis Data Firehose 传输

流上设置的时间缓冲间隔显示在 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 'my-bucket' --prefix 'firehose/'

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

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

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

Amazon S3 对象中的数据以 gzip 格式压缩。您可以使用以下 Unix 命令检查命令行中的原始数据：

```
zcat testfile.gz
```

与订阅的跨账户日志数据共享

您可以与不同 AWS 账户的所有者合作，并在您的 AWS 资源 (例如 Amazon Kinesis 流) 上接收他们的日志事件 (这称为“跨账户数据共享”)。例如，此日志事件数据可从集中的 Amazon Kinesis 流中读取以执行自定义处理和分析。自定义处理在您跨多个账户协作和分析数据时特别有用。例如，某家公司的信息安全组可能需要分析数据以进行实时入侵检测或查找异常行为，以便能对公司所有部门的账户进行审核 (通过收集各账户的联合生产日志进行集中处理)。可以汇编这些账户中的事件数据的实时流并将其传递到信息安全组，这些组可使用 Kinesis 将数据附加到现有安全分析系统。

Kinesis 流目前是支持作为跨账户订阅目标的唯一资源。

要跨账户共享日志数据，您需要建立日志数据发送者和接收者：

- 日志数据发送者 - 从接收者获取目标信息并告知 CloudWatch Logs 发送者已准备好将其日志事件发送到指定目标。在本部分剩余的过程中，日志数据发送者显示为虚构的 AWS 账号 111111111111。
- 日志数据接收者 - 设置封装 Kinesis 流的目标并告知 CloudWatch Logs 接收者希望接收日志数据。接收者随后与发送者共享与接收者的目标有关的信息。在本部分剩余的过程中，日志数据接收者显示为虚构的 AWS 账号 999999999999。

要开始接收来自跨账户用户的日志事件，日志数据接收者应首先创建一个 CloudWatch Logs 目标。每个目标都包含以下键元素：

目标名称

您要创建的目标的名称。

目标 ARN

您要用作订阅源目标的 AWS 资源的 Amazon 资源名称 (ARN)。

角色 ARN

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

访问策略

一个 IAM 策略文档 (采用 JSON 格式，使用 IAM 策略语法编写)，用于管理有权对您的目标进行写入的用户组。

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

主题

创建目标

此过程中的步骤需要在日志数据接收者账户中完成。在本示例中，日志数据接收者账户具有 AWS 账户 ID 999999999999，而日志数据发送者 AWS 账户 ID 为 111111111111。

此示例使用名为 RecipientStream 的 Kinesis 流创建目标，并创建支持 CloudWatch Logs 将数据写入目标的角色。

创建目标

1. 在 Kinesis 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 Kinesis 流变为活动状态。您可使用 aws kinesis describe-stream 命令检查 StreamDescription.StreamStatus 属性。此外，请记下 StreamDescription.StreamARN 值，因为该值稍后将传递到 CloudWatch Logs：

```
aws kinesis describe-stream --stream-name "RecipientStream"
{
  "StreamDescription": {
```

```
    "StreamStatus": "ACTIVE",
    "StreamName": "RecipientStream",
    "StreamARN": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "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，该角色将向 CloudWatch Logs 授予将数据放入 Kinesis 流的权限。首先，您需要在文件 ~/TrustPolicyForCWL.json 中创建信任策略。使用文本编辑器创建此策略文件，请勿使用 IAM 控制台来创建。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.us-east-1.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

4. 使用 aws iam create-role 命令创建 IAM 角色，并指定信任策略文件。记下返回的 Role.Arn 值，因为该值稍后也将传递到 CloudWatch Logs：

```
aws iam create-role \
      --role-name CWLtoKinesisRole \
      --assume-role-policy-document file://~/TrustPolicyForCWL.json

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.us-east-1.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 Logs 可对您的账户执行的操作。首先，您需要使用文本编辑器在文件 ~/PermissionsForCWL.json 中创建权限策略：

```
{
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::999999999999:role/CWLtoKinesisRole"
    }
  ]
}
```

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 Kinesis 流进入活动状态并且您已创建 IAM 角色后，您便可以创建 CloudWatch Logs 目标。

   a. 此步骤不会将访问策略与您的目标关联，它只是完成目标创建的两个步骤中的第一个步骤。记下负载中返回的 DestinationArn：

```
aws logs put-destination \
    --destination-name "testDestination" \
    --target-arn "arn:aws:kinesis:us-east-1:999999999999: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。此策略使得日志数据发送者账户 (111111111111) 可以访问日志数据接收者账户 (999999999999) 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：

```
{
  "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. 这将创建一个策略，该策略定义了对目标具有写入权限的人。此策略必须指定 logs:PutSubscriptionFilter 操作才能访问目标。跨账户用户将使用 PutSubscriptionFilter 操作向目标发送日志事件：

```
aws logs put-destination-policy \
```

```
    --destination-name "testDestination" \
    --access-policy file://~/AccessPolicy.json
```

此访问策略允许 ID 为 111111111111 的 AWS 账户的根用户针对 ARN 为 arn:aws:logs:us-east-1:999999999999:destination:testDestination 的目标调用 PutSubscriptionFilter。任何其他用户针对此目标调用 PutSubscriptionFilter 的尝试都会被拒绝。

要根据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。

创建订阅筛选器

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器。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。

在以下示例中，订阅筛选器与包含 AWS CloudTrail 事件的日志组关联，以便将由“根”AWS 凭证记录的每个活动传输到您在上面创建的目标 (用于封装名为“RecipientStream”的 Kinesis 流) 。有关如何将 AWS CloudTrail 事件发送到 CloudWatch Logs 的更多信息，请参阅 AWS CloudTrail User Guide 中的将 CloudTrail 事件发送到 CloudWatch Logs。

```
aws logs put-subscription-filter \
    --log-group-name "CloudTrail" \
    --filter-name "RecipientStream" \
    --filter-pattern "{$.userIdentity.type = Root}" \
    --destination-arn "arn:aws:logs:us-east-1:999999999999:destination:testDestination"
```

日志组和目标必须位于同一 AWS 区域内。但是，目标可指向位于不同区域的 AWS 资源，如 Kinesis 流。

Note

与订阅示例 使用订阅实时处理日志数据 (p. 51) 不同，在此示例中，您不必提供角色 ARN。这是因为在对 Kinesis 流进行写入时需要角色 ARN 来进行模拟，ARN 角色已由目标所有者在创建目标时提供。

验证日志事件的流动

在创建订阅筛选器后，CloudWatch Logs 将与筛选器模式匹配的所有传入日志事件转发至名为“RecipientStream”的目标流中封装的 Kinesis 流。目标所有者可使用 aws kinesis get-shard-iterator 命令抓取 Kinesis 分片并使用 aws kinesis get-records 命令提取一些 Kinesis 记录，以验证此事件是否正在发生：

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

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

aws kinesis get-records \
      --limit 10 \
      --shard-iterator
```

```
      "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+OIK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
```

Note

您可能需要在 Kinesis 开始返回数据前几分钟重新运行 get-records 命令。

您将看到包含一组 Kinesis 记录的响应。Kinesis 记录中的数据属性使用 gzip 格式进行 Base64 编码和压缩。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",
    "logStream": "111111111111_CloudTrail_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器名称的列表。

messageType

数据消息将使用“DATA_MESSAGE”类型。有时候，CloudWatch Logs 可能发出类型为“CONTROL_MESSAGE”的 Kinesis 记录，这主要是为了检查目标是否可访问。

logEvents

表示为一组日志事件记录的实际日志数据。ID 属性是每个日志事件的唯一标识符。

在运行时修改目标成员资格

您可能遇到必须在您拥有的目标中添加或删除某些用户的成员资格的情况。您可通过新访问策略对您的目标使用 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 222222222222。

1. 提取当前与目标 testDestination 关联的策略并记下 AccessPolicy：

```
aws logs describe-destinations \
    --destination-name-prefix "testDestination"

{
 "Destinations": [
   {
     "DestinationName": "testDestination",
     "RoleArn": "arn:aws:iam::123456789012:role/CWLtoKinesisRole",
     "DestinationArn": "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
     "TargetArn": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream",
     "AccessPolicy": "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Sid
\": \"\", \"Effect\": \"Allow\", \"Principal\": {\"AWS\": \"234567890123\"},
 \"Action\": \"logs:PutSubscriptionFilter\", \"Resource\": \"arn:aws:logs:us-
east-1:123456789012:destination:testDestination\"}] }"
   }
 ]
}
```

2. 更新该策略以反映已阻止账户 111111111111，并且已启用账户 222222222222。将此策略放入 ~/NewAccessPolicy.json 文件：

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "222222222222"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-east-1:123456789012:destination:testDestination"
    }
  ]
}
```

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

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

这将最终禁用账户 ID 111111111111 中的日志事件。一旦账户 222222222222 的所有者使用 PutSubscriptionFilter 创建订阅筛选器，账户 ID 222222222222 中的日志事件就会立即开始流向目标。

将日志数据导出至 Amazon S3

您可以将日志数据从您的日志组导出至 Amazon S3 存储桶，然后使用此数据进行自定义处理和分析，或将其载入到其他系统。

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 Amazon S3 存储桶中，并定义 Amazon S3 生命周期规则以自动地存档或删除已导出的文件。

可将多个日志组或多个时间范围的日志导出至同一个 S3 存储桶中。要分开每个导出任务的日志数据，您可以指定一个前缀，以便用作所有导出对象的 Amazon S3 键前缀。

日志数据可能需要长达 12 小时才能用于导出。有关日志数据接近实时的分析，请参阅 使用订阅实时处理日志数据 (p. 51)。

内容

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。此日志组中的日志数据将导出至指定的 Amazon S3 存储桶中。

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。 日志组中该时间之后获得的所有日志事件都会被导出。

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 Amazon S3 存储桶的名称。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 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

下面的示例使用 Amazon CloudWatch 控制台将所有数据从名为“my-log-group”的 Amazon CloudWatch Logs 日志组导出到名为“my-exported-logs”的 Amazon S3 存储桶中。

步骤 1：创建 Amazon S3 存储桶

我们建议您使用专为 CloudWatch Logs 创建的存储桶。 但是，如果要使用现有存储桶，请跳至第 2 步。

> Note
>
> Amazon S3 存储桶必须与要导出的日志数据处于同一个区域内。CloudWatch Logs 不支持将数据导出至其他区域中的 Amazon S3 存储桶。

创建 Amazon S3 存储桶

1. 通过以下网址打开 Amazon S3 控制台：https://console.amazonaws.cn/s3/。
2. 如果需要，可以更改区域。从导航栏中，选择您的 CloudWatch Logs 所在的区域。
3. 选择 Create Bucket。
4. 为 Bucket Name 键入存储桶的名称。
5. 为 Region 选择您的 CloudWatch Logs 数据所在的地区。
6. 选择 Create。

第 2 步：在 Amazon S3 存储桶上设置权限

默认情况下，所有 Amazon S3 存储桶和对象都是私有的。仅资源所有者 (创建了存储桶的 AWS 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。

要设置 Amazon S3 存储段上的权限

1. 在 Amazon S3 控制台中，选择您在第 1 步中创建的存储桶。
2. 选择 Permissions、Add bucket policy。
3. 在 Bucket Policy Editor 对话框，添加以下策略，将 `Resource` 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终端节点。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs",
          "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
      },
      {
          "Action": "s3:PutObject" ,
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs/*",
          "Condition": { "StringEquals": { "s3:x-amz-acl": "bucket-owner-full-
control" } },
          "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
      }
    ]
}
```

4. 选择 Save，将您刚添加的策略设置为存储桶上的访问策略。此策略可让 CloudWatch Logs 将日志数据导出至您的 Amazon S3 存储桶中。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一个或多个策略，请将 CloudWatch Logs 访问的语句添加到这些策略。我们建议您评估生成的权限集，确保它们适合访问存储桶的用户。

第 3 步：创建导出任务

在本步骤中，您可创建导出任务以便从日志组中导出日志。

使用 CloudWatch 控制台将数据导出到 Amazon S3

1. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。
3. 在 Log Groups 屏幕上，选中日志组旁边的复选框，然后选择 Actions、Export data to Amazon S3。
4. 在 Export data to Amazon S3 屏幕的 Define data to export 下，使用 From 和 To 设置要导出的数据的时间范围。
5. 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，然后为 Stream prefix 键入日志流前缀。
6. 在 Choose S3 bucket 中，选择与 Amazon S3 存储桶关联的账户。
7. 为 S3 bucket name 选择 Amazon S3 存储桶。
8. 要分开每个导出任务的日志数据，您可以指定一个 Amazon S3 前缀，以便用作所有导出对象的 Amazon S3 键前缀。选择 Advanced，然后为 S3 Bucket prefix 键入存储桶前缀。
9. 选择 Export data，将日志数据导出到 Amazon S3。
10. 要查看您导出到 Amazon S3 的日志数据的状态，请选择 Actions、View all exports to Amazon S3。

使用 AWS CLI 将数据导出到 Amazon S3

下面的示例使用导出任务将所有数据从名为“my-log-group”的 CloudWatch Logs 日志组导出到名为“my-exported-logs”的 Amazon S3 存储桶中。此示例假定您已创建了一个名为“my-log-group”的日志组。

步骤 1：创建 Amazon S3 存储桶

我们建议您使用专为 CloudWatch Logs 创建的存储桶。 但是，如果要使用现有存储桶，请跳至第 2 步。

> **Note**
>
> Amazon S3 存储桶必须与要导出的日志数据处于同一个区域内。CloudWatch Logs 不支持将数据导出至其他区域中的 Amazon S3 存储桶。

使用 AWS CLI 创建 Amazon S3 存储桶

在命令提示符处，运行以下 create-bucket 命令，其中 `LocationConstraint` 是您要将日志数据导出到的区域：

```
aws s3api create-bucket --bucket my-exported-logs --create-bucket-configuration
 LocationConstraint=us-west-2
```

下面是示例输出：

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

第 2 步：在 Amazon S3 存储桶上设置权限

默认情况下，所有 Amazon S3 存储桶和对象都是私有的。仅资源所有者 (创建了存储桶的 AWS 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。

要设置 Amazon S3 存储段上的权限

1. 创建一个名为 policy.json 的文件并添加以下访问策略，将 `Resource` 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终端节点。使用文本编辑器以创建此策略文件。请勿使用 IAM 控制台。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "s3:GetBucketAcl",
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::my-exported-logs",
            "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
        },
        {
            "Action": "s3:PutObject" ,
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::my-exported-logs/*",
            "Condition": { "StringEquals": { "s3:x-amz-acl": "bucket-owner-full-
control" } },
            "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
        }
    ]
}
```

2. 使用 put-bucket-policy 命令将您刚添加的策略设置为存储桶上的访问策略。此策略可让 CloudWatch Logs 将日志数据导出至您的 Amazon S3 存储桶中。存储桶拥有者将对所有导出的对象拥有完全权限。

```
aws s3api put-bucket-policy --bucket my-exported-logs --policy file://policy.json
```

Warning

如果现有存储桶已附加一个或多个策略，请将 CloudWatch Logs 访问的语句添加到这些策略。我们建议您评估生成的权限集，确保它们适合访问存储桶的用户。

第 3 步：创建导出任务

创建从日志组导出日志的导出任务后，导出任务可能需要几秒到几小时的时间才能完成，具体取决于要导出的数据大小。

使用 AWS CLI 创建导出任务

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 --task-name "my-log-group-09-10-2015" --log-group-name "my-
log-group" --from 1441490400000 --to 1441494000000 --destination "my-exported-logs" --
destination-prefix "export-task-output"
```

下面是示例输出：

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

第 4 步：说明导出任务

创建导出任务后，您可以获取任务的当前状态。

使用 AWS CLI 描述导出任务

在命令提示符处，使用以下 describe-export-tasks 命令：

```
aws logs 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",
      "To": 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 describe-export-tasks --status-code "FAILED"
```

下面是示例输出：

```
{
   "ExportTasks": [
   {
      "Destination": "my-exported-logs",
      "DestinationPrefix": "export-task-output",
      "ExecutionInfo": {
         "CompletionTime": 1441498600000
         "CreationTime": 1441495400000
      },
      "From": 1441490400000,
      "LogGroupName": "my-log-group",
      "Status": {
         "Code": "FAILED",
         "Message": "FAILED"
```

```
      },
      "TaskId": "cda45419-90ea-4db5-9833-aade86253e66",
      "TaskName": "my-log-group-09-10-2015",
      "To": 1441494000000
   }]
}
```

第 5 步：取消导出任务

您可以取消处于 PENDING (等待完成) 或 RUNNING (正在运行) 状态的导出任务。

使用 AWS CLI 取消导出任务

在命令提示符处，使用以下 cancel-export-task 命令：

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

请注意，您可以使用 describe-export-tasks 命令验证任务是否已成功取消。

将 CloudWatch Logs 数据流式传输到 Amazon Elasticsearch Service

您可以将 CloudWatch Logs 日志组配置为通过 CloudWatch Logs 订阅将其收到的数据实时流式传输到 Amazon Elasticsearch Service (Amazon ES) 群集中。有关更多信息，请参阅 使用订阅实时处理日志数据 (p. 51)。

Note

将大量 CloudWatch Logs 数据流式传输到 Amazon ES 可能会产生较高的使用费。我们建议您监控 AWS 账单以帮助避免产生超出预期的费用。有关更多信息，请参阅使用 CloudWatch 监控您的估计费用。

先决条件

在开始之前，请创建 Amazon ES 域。Amazon ES 域可能具有公有访问权限或 VPC 访问权限，但是您无法在创建该域后修改访问权限的类型。稍后您可能需要检查 Amazon ES 域设置，并基于群集将处理的数据量修改群集配置。

有关 Amazon ES 的更多信息，请参阅 Amazon Elasticsearch Service 开发人员指南。

创建 Amazon ES 域

在命令提示符处，使用以下 create-elasticsearch-domain 命令：

```
aws es create-elasticsearch-domain --domain-name my-domain
```

将日志组订阅到 Amazon ES

您可以使用 CloudWatch 控制台将日志组订阅到 Amazon ES。

将日志组订阅到 Amazon ES

1. 通过以下网址打开 CloudWatch 控制台：https://console.amazonaws.cn/cloudwatch/。
2. 在导航窗格中，选择 Logs。
3. 选择要订阅的日志组。
4. 选择 Actions、Stream to Amazon Elasticsearch Service。
5. 在 Start Streaming to Amazon Elasticsearch Service 屏幕上，为 Amazon ES cluster 选择您在上一步中创建的群集，然后选择 Next。
6. 在 Lambda Function 下，为 Lambda IAM Execution Role 选择 Lambda 在执行对 Amazon ES 的调用时应使用的 IAM 角色，然后选择 Next。

   您选择的 IAM 角色必须满足以下要求：

   - 它在信任关系中必须具有 `lambda.amazonaws.com`。
   - 它必须包含以下策略：

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

- 如果目标 Amazon ES 域使用 VPC 访问权限，则该角色必须已附加 AWSLambdaVPCAccessExecutionRole 策略。此 Amazon 托管策略将为 Lambda 授予对客户的 VPC 的访问权限，从而使 Lambda 能够在 VPC 中写入 Amazon ES 终端节点。

7. 在 Configure Log Format and Filters 屏幕上，为 Log Format 选择日志格式。
8. 在 Subscription Filter Pattern 下，键入要在您的日志事件中查找的字词或模式。这可确保只将您感兴趣的数据发送到 Amazon ES 群集。有关更多信息，请参阅 搜索和筛选日志数据 (p. 35)。
9. (可选) 在 Select Log Data to Test 下，选择一个日志流，然后单击 Test Pattern 以确认搜索筛选器是否会返回您期望的结果。
10. 选择 Next，然后在 Review & Start Streaming to Amazon Elasticsearch Service 屏幕上，选择 Start Streaming。

Amazon CloudWatch Logs 的身份验证和访问控制

访问 Amazon CloudWatch Logs 时需要有 AWS 可以用来验证您的请求的凭证。这些凭证必须有权访问 AWS 资源，例如检索有关您的云资源的 CloudWatch Logs 数据。下面几节提供详细的信息来说明如何使用 AWS Identity and Access Management (IAM) 和 CloudWatch Logs 控制谁能访问您的资源，从而对这些资源进行保护：

身份验证

您可以下面任一类型的身份访问 AWS：

- AWS 账户根用户 – 注册 AWS 时，您需要提供与您的 AWS 账户关联的电子邮件地址和密码。这些是您的根凭证，它们提供对您所有 AWS 资源的完全访问权限。

  Important

  出于安全考虑，我们建议您仅使用根凭证创建管理员用户，此类用户是对您的 AWS 账户具有完全访问权限的 IAM 用户。随后，您可以使用此管理员用户来创建具有有限权限的其他 IAM 用户和角色。有关更多信息，请参阅 IAM 用户指南 中的 IAM 最佳实践和创建管理员用户和组。
- IAM 用户 – IAM 用户就是您的 AWS 账户中的一种身份，它具有特定的自定义权限 (例如，用于在 CloudWatch Logs 中查看指标的权限)。您可以使用 IAM 用户名和密码来登录以保护 AWS 网页，如 AWS 管理控制台、AWS 开发论坛或 AWS Support Center。

  除了用户名和密码之外，您还可以为每个用户生成访问密钥。在通过多个软件开发工具包之一或使用 AWS Command Line Interface (AWS CLI) 以编程方式访问 AWS 服务时，可以使用这些密钥。SDK 和 CLI 工具使用访问密钥对您的请求进行加密签名。如果您不使用 AWS 工具，则必须自行对请求签名。CloudWatch Logs supports 签名版本 4，后者是一种用于对入站 API 请求进行身份验证的协议。有关验证请求的更多信息，请参阅 AWS General Reference 中的签名版本 4 签名流程。

- IAM 角色 – IAM 角色是可在账户中创建的另一种具有特定权限的 IAM 身份。它类似于 IAM 用户，但未与特定人员关联。利用 IAM 角色，您可以获得可用于访问 AWS 服务和资源的临时访问密钥。具有临时凭证的 IAM 角色在以下情况下很有用：

  - 联合身份用户访问 – 您可以不创建 IAM 用户，而是使用来自 AWS Directory Service、您的企业用户目录或 Web 身份提供商的既有用户身份。他们被称为联合身份用户。在通过身份提供商请求访问权限时，AWS 将为联合身份用户分配角色。有关联合身份用户的更多信息，请参阅 IAM 用户指南 中的联合身份用户和角色。

  - 跨账户访问 – 可以使用您账户中的 IAM 角色向另一个 AWS 账户授予对您账户的资源的访问权限。有关示例，请参阅 &guide-iam-user; 中的教程：使用 IAM 角色委派跨 AWS 账户的访问权限。

- AWS 服务访问 – 可以使用您账户中的 IAM 角色向 AWS 服务授予对您账户的资源的访问权限。例如，您可以创建一个角色，此角色允许 Amazon Redshift 代表您访问 Amazon S3 存储桶，然后将存储在该存储桶中的数据加载到 Amazon Redshift 群集中。有关更多信息，请参阅 IAM 用户指南 中的创建向 AWS 服务委派权限的角色。

- 在 Amazon EC2 上运行的应用程序 – 您不用将访问密钥存储在 EC2 实例中以供实例上运行的应用程序使用并发出 AWS API 请求，而是可以使用 IAM 角色管理这些应用程序的临时凭证。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM 用户指南 中的对 Amazon EC2 上的应用程序使用角色。

访问控制

您可以使用有效的凭证来对自己的请求进行身份验证，但您还必须拥有权限才能创建或访问 CloudWatch Logs 资源。例如，您必须拥有创建日志流、创建日志组和执行其他操作的权限。

下面几节介绍如何管理 CloudWatch Logs 的权限。我们建议您先阅读概述。

管理您的 CloudWatch Logs 资源的访问权限概述

每个 AWS 资源都归某个 AWS 账户所有，创建和访问资源的权限由权限策略进行管理。账户管理员可以向 IAM 身份（即：用户、组和角色）挂载权限策略，某些服务（如 AWS Lambda）也支持向资源挂载权限策略。

> Note
>
> 账户管理员（或管理员用户）是具有管理员权限的用户。有关更多信息，请参阅 IAM 用户指南 中的 IAM 最佳实践。

在授予权限时，您要决定谁获得权限，获得对哪些资源的权限，以及您允许对这些资源执行的具体操作。

主题

CloudWatch Logs 资源和操作

在 CloudWatch Logs 中，主资源是日志组、日志流和目标。CloudWatch Logs 不支持子资源 (与主资源一起使用的其他资源)。

这些资源和子资源具有与其关联的唯一 Amazon 资源名称 (ARN)，如下表所示。

资源类型	ARN 格式
日志组	arn:aws:logs:*region*:*account-id*:log-group:*log_group_name*
日志流	arn:aws:logs:*region*:*account-id*:log-group:*log_group_name*:log-stream:*log-stream-name*
目的地	arn:aws:logs:*region*:*account-id*:destination:*destination_name*

有关 ARN 的更多信息，请参阅 IAM 用户指南 中的 ARN。有关 CloudWatch Logs ARN 的信息，请参阅 Amazon Web Services 一般参考 中的 Amazon 资源名称 (ARN) 和 AWS 服务命名空间。要查看包含 CloudWatch Logs 的策略的示例，请参阅为 CloudWatch Logs 使用基于身份的策略 (IAM 策略) (p. 80)。

CloudWatch Logs 提供一组操作用来处理 CloudWatch Logs 资源。有关可用操作的列表，请参阅 CloudWatch Logs 权限参考 (p. 84)。

了解资源所有权

AWS 账户对在该账户下创建的资源具有所有权，而无论创建资源的人员是谁。具体而言，资源所有者是对资源创建请求进行身份验证的委托人实体 (即根账户、IAM 用户或 IAM 角色) 的 AWS 账户。以下示例说明了它的工作原理：

- 如果您使用 AWS 账户的根账户凭证创建日志组，则您的 AWS 账户即为该 CloudWatch Logs 资源的所有者。
- 如果您在您的 AWS 账户中创建 IAM 用户并对该用户授予创建 CloudWatch Logs 资源的权限，则该用户可以创建 CloudWatch Logs 资源。但是，该用户所属的 AWS 账户拥有这些 CloudWatch Logs 资源。
- 如果您在您的 AWS 账户中创建具有创建 CloudWatch Logs 资源的权限的 IAM 角色，则能够担任该角色的任何人都可以创建 CloudWatch Logs 资源。该角色所属的 AWS 账户拥有这些 CloudWatch Logs 资源。

管理对资源的访问

权限策略 规定谁可以访问哪些内容。下一节介绍创建权限策略时的可用选项。

> Note
>
> 本节讨论如何在 CloudWatch Logs 范围内使用 IAM。它不提供有关 IAM 服务的详细信息。有关完整的 IAM 文档，请参阅什么是 IAM？（在 IAM 用户指南 中）。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。

挂载到 IAM 身份的策略称作基于身份 的策略 (IAM 策略)，而挂载到资源的策略称作基于资源 的策略。CloudWatch Logs 支持基于身份的策略以及适用于目标的基于资源的策略，该策略用于启用跨账户订阅。有关更多信息，请参阅 与订阅的跨账户日志数据共享 (p. 61)。

主题

- 基于身份的策略 (IAM 策略) (p. 78)
- 基于资源的策略 (p. 79)

基于身份的策略 (IAM 策略)

您可以向 IAM 身份挂载策略。例如，您可以执行以下操作：

- 将权限策略附加到您的账户中的用户或组 - 要授予在 CloudWatch Logs 控制台中查看日志的用户权限，您可以将权限策略附加到用户或用户所属的组。
- 向角色挂载权限策略 (授予跨账户权限) - 您可以向 IAM 角色挂载基于身份的权限策略，以授予跨账户的权限。例如，账户 A 中的管理员可以创建一个角色，以向其他 AWS 账户 (如账户 B) 或某项 AWS 服务授予跨账户权限，如下所述：
  1. 账户 A 管理员创建一个 IAM 角色，向该角色挂载授权其访问账户 A 中资源的权限策略。
  2. 账户 A 管理员可以向将账户 B 标识为能够担任该角色的委托人的角色挂载信任策略。
  3. 之后，账户 B 管理员可以委派权限，指定账户 B 中的任何用户担任该角色。这样，账户 B 中的用户就可以创建或访问账户 A 中的资源了。如果您需要授予 AWS 服务权限来担任该角色，则信任策略中的委托人也可以是 AWS 服务委托人。

  有关使用 IAM 委派权限的更多信息，请参阅 IAM 用户指南 中的访问权限管理。

下面是为 us-east-1 中的所有资源上的 `logs:PutLogEvents`、`logs:CreateLogGroup` 和 `logs:CreateLogStream` 操作授予权限的示例策略。对于日志组，CloudWatch Logs 支持对某些 API 操作使用资源 ARN (也称为资源级权限) 标识特定资源。如果您希望包含所有日志组，则必须指定通配符 (*)。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "234567890123"
      },
      "Action" : [
          "logs:PutLogEvents",
          "logs:CreateLogGroup",
          "logs:CreateLogStream",
      "Resource" : "arn:aws:logs:us-east-1:*:*"
    }
  ]
}
```

有关将基于身份的策略用于 CloudWatch Logs 的更多信息，请参阅为 CloudWatch Logs 使用基于身份的策略 (IAM 策略) (p. 80)。有关用户、组、角色和权限的更多信息，请参阅 IAM 用户指南 中的身份 (用户、组和角色) 。

基于资源的策略

CloudWatch Logs 支持适用于目标的基于资源的策略，您可以使用该策略启用跨账户订阅。有关更多信息，请参阅 创建目标 (p. 62)。可使用 PutDestination API 创建目标，您可使用 PutDestination API 向目标添加资源策略。以下示例允许账户 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`。

```
{
  "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、效果和委托人

对于每种 CloudWatch Logs 资源，该服务都定义了一组 API 操作。为授予这些 API 操作的权限，CloudWatch Logs 定义了一组您可以在策略中指定的操作。某些 API 操作可能需要多个操作的权限才能执行 API 操作。有关资源和 API 操作的更多信息，请参阅 CloudWatch Logs 资源和操作 (p. 77) 和 CloudWatch Logs 权限参考 (p. 84)。

以下是基本的策略元素：

- Resource - 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 Logs 资源和操作 (p. 77)。
- Action – 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。
- Effect - 用于指定当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。
- Principal - 在基于身份的策略 (IAM 策略) 中，附加了策略的用户是隐式委托人。对于基于资源的策略，您可以指定要接收权限的用户、账户、服务或其他实体 (仅适用于基于资源的策略)。CloudWatch Logs 支持适用于目标的基于资源的策略。

要了解有关 IAM 策略语法和说明的更多信息，请参阅 IAM 用户指南 中的 AWS IAM 策略参考。

有关显示所有 CloudWatch Logs API 操作及其适用资源的表，请参阅 CloudWatch Logs 权限参考 (p. 84)。

在策略中指定条件

当您授予权限时，可使用访问策略语言来指定规定策略何时生效的条件。例如，您可能希望策略仅在特定日期后应用。有关使用策略语言指定条件的更多信息，请参阅 IAM 用户指南 中的条件。

要表示条件，您可以使用预定义的条件键。有关每个 AWS 服务支持的上下文键列表以及 AWS 范围的策略键列表，请参阅 IAM 用户指南 中的 AWS 服务操作和条件上下文键和全局和 IAM 条件上下文键。

为 CloudWatch Logs 使用基于身份的策略 (IAM 策略)

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) 挂载权限策略。

> **Important**
>
> 我们建议您首先阅读以下介绍性主题，这些主题讲解了管理 CloudWatch Logs 资源访问的基本概念和选项。有关更多信息，请参阅 管理您的 CloudWatch Logs 资源的访问权限概述 (p. 77)。

本主题包含以下内容：

- 使用 CloudWatch 控制台所需的权限 (p. 81)
- 适用于 CloudWatch Logs 的 AWS 托管 (预定义) 策略 (p. 82)
- 客户托管策略示例 (p. 83)

下面是权限策略的示例：

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

本策略具有一个语句，该语句授予了创建日志组和日志流、将事件上传到日志流和列出有关日志流的详细信息的权限。

`Resource` 值结尾的通配符 (*) 表示一个语句，该语句授予了对任何日志组执行 `logs:CreateLogGroup`、`logs:CreateLogStream`、`logs:PutLogEvents` 和 `logs:DescribeLogStreams` 操作的权限。要将此权限限制到特定日志组，请使用将资源 ARN 中的通配符 (*) 替换为特定日志组 ARN。有关 IAM 策略语句中的各个部分的更多信息，请参阅 IAM 用户指南 中的 IAM 策略元素参考。有关显示所有 CloudWatch Logs 操作的列表，请参阅 CloudWatch Logs 权限参考 (p. 84)。

使用 CloudWatch 控制台所需的权限

用户若要能够使用 CloudWatch 控制台中的 CloudWatch Logs，则必须拥有一组可在其 AWS 账户中描述其他 AWS 资源的最低权限。要使用 CloudWatch 控制台中的 CloudWatch Logs，您必须拥有来自以下服务的权限：

- CloudWatch
- CloudWatch Logs
- Amazon ES
- IAM
- Kinesis
- Lambda
- Amazon S3

如果创建比必需的最低权限更为严格的 IAM 策略，对于附加了该 IAM 策略的用户，控制台将无法按预期正常运行。为确保这些用户仍可使用 CloudWatch 控制台，同时向用户附加 `CloudWatchReadOnlyAccess` 托管策略，请参阅 适用于 CloudWatch Logs 的 AWS 托管 (预定义) 策略 (p. 82)。

对于只需要调用 AWS CLI 或 CloudWatch Logs API 的用户，您无需为其提供最低控制台权限。

对于未使用控制台管理日志订阅的用户，使用 CloudWatch 控制台所需的完整权限集为：

- cloudwatch:getMetricData
- cloudwatch:listMetrics
- logs:cancelExportTask
- logs:createExportTask

- logs:createLogGroup
- logs:createLogStream
- logs:deleteLogGroup
- logs:deleteLogStream
- logs:deleteMetricFilter
- logs:deleteRetentionPolicy
- logs:deleteSubscriptionFilter
- logs:describeExportTasks
- logs:describeLogGroups
- logs:describeLogStreams
- logs:describeMetricFilters
- logs:describeSubscriptionFilters
- logs:filterLogEvents
- logs:getLogEvents
- logs:putMetricFilter
- logs:putRetentionPolicy
- logs:putSubscriptionFilter
- logs:testMetricFilter

对于同时使用控制台来管理日志订阅的用户，还需要以下权限：

- es:describeElasticsearchDomain
- es:listDomainNames
- iam:attachRolePolicy
- iam:createRole
- iam:getPolicy
- iam:getPolicyVersion
- iam:getRole
- iam:listAttachedRolePolicies
- iam:listRoles
- kinesis:describeStreams
- kinesis:listStreams
- lambda:addPermission
- lambda:createFunction
- lambda:getFunctionConfiguration
- lambda:listAliases
- lambda:listFunctions
- lambda:listVersionsByFunction
- lambda:removePermission
- s3:listBuckets

适用于 CloudWatch Logs 的 AWS 托管 (预定义) 策略

AWS 通过提供由 AWS 创建和管理的独立 IAM 策略来解决很多常用案例。托管策略可授予常用案例的必要权限，因此，您可以免去调查都需要哪些权限的工作。有关更多信息，请参阅 IAM 用户指南 中的 AWS 托管策略。

以下 AWS 托管策略 (您可以将它们挂载到自己账户中的用户) 是特定于 CloudWatch Logs 的：

- CloudWatchLogsFullAccess - 授予对 CloudWatch Logs 的完全访问权限。
- CloudWatchLogsReadOnlyAccess - 授予对 CloudWatch Logs 的只读访问权限。

> **Note**
>
> 您可以通过登录到 IAM 控制台并在该控制台中搜索特定策略来查看这些权限策略。

此外，您还可以创建自己的自定义 IAM 策略，以授予 CloudWatch Logs 操作和资源的相关权限。您可以将这些自定义策略挂载到需要这些权限的 IAM 用户或组。

客户托管策略示例

本节的用户策略示例介绍如何授予各 CloudWatch Logs 操作的权限。当您使用 CloudWatch Logs API、AWS 软件开发工具包或 AWS CLI 时，可以使用这些策略。

示例

示例 1：允许对 CloudWatch Logs 进行完全访问

以下策略允许用户访问所有 CloudWatch Logs 操作。

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

示例 2：允许对 CloudWatch Logs 进行只读访问

以下策略允许用户对 CloudWatch Logs 数据进行只读访问。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "logs:Describe*",
            "logs:Get*",
            "logs:TestMetricFilter",
            "logs:FilterLogEvents"
         ],
         "Effect":"Allow",
         "Resource":"*"
      }
   ]
}
```

CloudWatch Logs 权限参考

在设置 访问控制 (p. 77) 和编写您可挂载到 IAM 身份的权限策略 (基于身份的策略) 时，可以使用下表作为参考。此表列出每个 CloudWatch Logs API 操作及您可授予执行该操作的权限的对应操作。可在策略的 `Action` 字段中指定操作，在策略的 `Resource` 字段中指定通配符 (*) 作为资源值。

您可以在 CloudWatch Logs 策略中使用 AWS 范围的条件键来表达条件。有关 AWS 范围的键的完整列表，请参阅 IAM 用户指南 中的 AWS 全局和 IAM 条件上下文键。

> Note
>
> 要指定操作，请在 API 操作名称之前使用 `logs:` 前缀。例如：`logs:CreateLogGroup`、`logs:CreateLogStream` 或 `logs:*` (针对所有 CloudWatch Logs 操作)。

CloudWatch Logs API 操作和必需的操作权限

CloudWatch Logs API 操作	所需权限 (API 操作)
CancelExportTask	`logs:CancelExportTask` 要求取消待处理或正在运行的导出任务。
CreateExportTask	`logs:CreateExportTask` 要求将数据从日志组导出到 Amazon S3 存储桶。
CreateLogGroup	`logs:CreateLogGroup` 要求创建新的日志组。
CreateLogStream	`logs:CreateLogStream` 要求在日志组中创建新的日志流。
DeleteDestination	`logs:DeleteDestination` 要求删除日志目标并对其禁用所有订阅筛选器。
DeleteLogGroup	`logs:DeleteLogGroup` 要求删除日志组和所有关联的存档日志事件。
DeleteLogStream	`logs:DeleteLogStream` 要求删除日志流和所有关联的存档日志事件。
DeleteMetricFilter	`logs:DeleteMetricFilter` 要求删除与日志组关联的指标筛选器。
DeleteRetentionPolicy	`logs:DeleteRetentionPolicy` 要求删除日志组的保留策略。
DeleteSubscriptionFilter	`logs:DeleteSubscriptionFilter` 要求删除与日志组关联的订阅筛选器。
DescribeDestinations	`logs:DescribeDestinations` 要求查看与账户关联的所有目标。

CloudWatch Logs API 操作	所需权限 (API 操作)
DescribeExportTasks	`logs:DescribeExportTasks` 要求查看与账户关联的所有导出任务。
DescribeLogGroups	`logs:DescribeLogGroups` 要求查看与账户关联的所有日志组。
DescribeLogStreams	`logs:DescribeLogStreams` 要求查看与日志组关联的所有日志流。
DescribeMetricFilters	`logs:DescribeMetricFilters` 要求查看与日志组关联的所有指标。
DescribeSubscriptionFilters	`logs:DescribeSubscriptionFilters` 要求查看与日志组关联的所有订阅筛选器。
FilterLogEvents	`logs:FilterLogEvents` 要求按照日志组筛选器模式对日志事件进行排序。
GetLogEvents	`logs:GetLogEvents` 要求从日志流中检索日志事件。
ListTagsLogGroup	`logs:ListTagsLogGroup` 要求列出与日志组关联的标签。
PutDestination	`logs:PutDestination` 要求创建或更新目标日志流 (例如，Kinesis 流)。
PutDestinationPolicy	`logs:PutDestinationPolicy` 要求创建或更新与现有日志目标关联的访问策略。
PutLogEvents	`logs:PutLogEvents` 要求将一批日志事件上传到日志流。
PutMetricFilter	`logs:PutMetricFilter` 要求创建或更新指标筛选器并将其与日志组关联。
PutRetentionPolicy	`logs:PutRetentionPolicy` 要求设置日志组中的日志事件的保存 (保留) 天数。
PutSubscriptionFilter	`logs:PutSubscriptionFilter` 要求创建或更新订阅筛选器并将其与日志组关联。
TagLogGroup	`logs:TagLogGroup` 要求添加或更新日志组标签。

CloudWatch Logs API 操作	所需权限 (API 操作)
TestMetricFilter	`logs:TestMetricFilter` 要求针对日志事件消息采样测试筛选器模式。

在 AWS CloudTrail 中记录 Amazon CloudWatch Logs API 调用

AWS CloudTrail 是一项服务，可用于捕获由您的 AWS 账户或代表您的 AWS 账户发出的 API 调用。这种信息经过收集后，写入存储在您指定的 Amazon S3 存储桶中的 CloudTrail 日志文件中。每当您使用 API、控制台或 AWS CLI 时，CloudTrail 就会记录 API 调用。通过使用由 CloudTrail 收集的信息，您可以确定发出了什么请求、发出请求的源 IP 地址、发出请求的人员以及发出请求的时间等。

要了解有关 CloudTrail 的更多信息 (包括如何对其进行配置和启用)，请参阅 AWS CloudTrail User Guide中的什么是 AWS CloudTrail。

内容

CloudTrail 中的 CloudWatch Logs 信息

如果启用 CloudTrail 日志记录，则对 API 操作的调用将捕获在 CloudTrail 日志文件中。每个日志文件条目都包含有关生成请求的人员的信息。例如，如果发出请求以创建 CloudWatch Logs 日志流 (`CreateLogStream`)，则 CloudTrail 会记录发出请求的人员或服务的用户身份。

日志条目中的用户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身份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 AWS CloudTrail User Guide 中的 CloudTrail userIdentity 元素。

CloudTrail 日志文件可以在 S3 存储桶中存储任意长时间，不过您也可以定义 Amazon S3 生命周期规则以自动存档或删除日志文件。默认情况下，将使用 Amazon S3 服务器端加密 (SSE) 对日志文件进行加密。

如果您需要获得 CloudTrail 日志文件传输的通知，则可以将 CloudTrail 配置为在传输新日志文件时发布 Amazon SNS 通知。有关更多信息，请参阅 AWS CloudTrail User Guide 中的为 CloudTrail 配置 Amazon SNS 通知。

您还可以将多个 AWS 区域和多个 AWS 账户中的 CloudTrail 日志文件聚合到单个 S3 存储桶中。有关更多信息，请参阅 AWS CloudTrail User Guide 中的从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件。

开启日志记录后，将在 CloudTrail 中为这些 CloudWatch Logs API 操作记录请求和响应元素：

- CancelExportTask
- CreateExportTask
- CreateLogGroup
- CreateLogStream
- DeleteDestination
- DeleteLogGroup

- DeleteLogStream
- DeleteMetricFilter
- DeleteRetentionPolicy
- DeleteSubscriptionFilter
- PutDestination
- PutDestinationPolicy
- PutMetricFilter
- PutRetentionPolicy
- PutSubscriptionFilter
- TestMetricFilter

将仅在 CloudTrail 中为以下 CloudWatch Logs API 操作记录请求元素：

- DescribeDestinations
- DescribeExportTasks
- DescribeLogGroups
- DescribeLogStreams
- DescribeMetricFilters
- DescribeSubscriptionFilters

不支持 `GetLogEvents`、`PutLogEvents` 和 `FilterLogEvents` CloudWatch Logs API 操作。

有关 CloudWatch Logs 操作的更多信息，请参阅 Amazon CloudWatch Logs API Reference。

了解日志文件条目

CloudTrail 日志文件包含一个或多个日志条目。每个条目列出了多个 JSON 格式的事件。一个日志条目表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。日志条目不是公用 API 调用的有序堆栈跟踪，因此它们不会以任何特定顺序显示。所有 API 操作的日志文件条目都类似于下面的示例。

以下日志文件条目显示某个用户调用了 CloudWatch Logs CreateExportTask 操作。

```
{
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/someuser",
            "accountId": "123456789012",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "someuser"
        },
        "eventTime": "2016-02-08T06:35:14Z",
        "eventSource": "logs.amazonaws.com",
        "eventName": "CreateExportTask",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
        "requestParameters": {
            "destination": "yourdestination",
            "logGroupName": "yourloggroup",
            "to": 123456789012,
```

```
            "from": 0,
            "taskName": "yourtask"
        },
        "responseElements": {
            "taskId": "15e5e534-9548-44ab-a221-64d9d2b27b9b"
        },
        "requestID": "1cd74c1c-ce2e-12e6-99a9-8dbb26bd06c9",
        "eventID": "fd072859-bd7c-4865-9e76-8e364e89307c",
        "eventType": "AwsApiCall",
        "apiVersion": "20140328",
        "recipientAccountId": "123456789012"
}
```

CloudWatch Logs 代理参考

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

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

代理配置文件

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

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

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

[logstream2]
...
```

state_file

指定状态文件的存储位置。

logging_config_file

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

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

```
cwlogs.push.watcher
```

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

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

[handlers]
keys=consoleHandler

[formatters]
keys=simpleFormatter

[logger_root]
level=INFO
handlers=consoleHandler

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

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

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

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

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

use_gzip_http_content_encoding

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

Note

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

log_group_name

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

log_stream_name

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

datetime_format

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

下面列出了常见 datetime_format 代码。您也可以使用 Python datetime.strptime() 支持的所有 datetime_format 代码。时区偏移量 (%z) 也受支持 (即使 Python 3.2 之前的版本都不支持它)，[+-]HHMM，不带冒号 (:)。有关更多信息，请参阅 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：使用 +HHMM 或 -HHMM 形式的 UTC 偏移量。+0000, -0400, +1030

示例格式：

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

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

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

time_zone

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

文件

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

file_fingerprint_lines

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

multi_line_start_pattern

指定用于识别日志消息开头的模式。日志消息由与模式匹配的行以及与模式不匹配的任何以下行组成。有效值是正则表达式或 {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 1000。

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 32768 字节。此大小的计算方式是 UTF-8 格式的所有事件消息之和加上代表每个日志事件的 26 字节。

结合使用 CloudWatch Logs 代理与 HTTP 代理

您可以将 CloudWatch Logs 代理与 HTTP 代理结合使用。

Note

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

结合使用 CloudWatch Logs 代理与 HTTP 代理

1. 执行以下任一操作：

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

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

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

为了维护在 EC2 实例上对 Amazon EC2 元数据服务的访问，请使用 --no-proxy 169.254.169.254 (推荐)。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的实例元数据和用户数据。

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

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

2. 重新启动代理以使更改生效：

```
sudo service awslogs restart
```

如果您使用的是 Amazon Linux 2，请使用以下命令重新启动代理。

```
sudo service awslogsd restart
```

划分 CloudWatch Logs 代理配置文件

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

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

重新启动代理以使更改生效：

```
sudo service awslogs restart
```

如果您使用的是 Amazon Linux 2，请使用以下命令重新启动代理。

```
sudo service awslogsd restart
```

CloudWatch Logs 代理常见问题

支持哪些类型的文件轮换？

支持以下文件轮换机制：

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

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

如何确定我使用的是哪个版本的代理？

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

日志条目如何转换为日志事件？

日志事件包含两个属性：事件发生时的时间戳，以及原始日志消息。默认情况下，以非空格字符开头的任何行都会使上一个日志消息结束 (如果存在)，并开始新的日志消息。要覆盖此行为，可以使用 multi_line_start_pattern，与模式匹配的任何行都开始新的日志消息。模式可以是任何正则表达式或“{datetime_format}”。例如，如果每个日志消息的第一行都包含类似于“2014-01-02T13:13:01Z”的时间戳，则 multi_line_start_pattern 可以设置为“\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}Z”。要简化配置，可以在指定 datetime_format 选项的情况下使用“{datetime_format}”变量。对于同一个示例，如果 datetime_format 设置为“%Y-%m-%dT%H:%M:%S%z”，则 multi_line_start_pattern 可以仅仅是“{datetime_format}”。

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

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

日志事件如何批处理？

满足以下任意条件时，表示批次已满并且将发布：

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

什么原因可能导致跳过或截断日志条目、日志事件或批次？

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

> **Note**
>
> 当跳过数据时，CloudWatch Logs 代理在其日志中写入一条警告。

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

停止代理会导致数据丢失/重复条目吗？

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

我可以将来自相同或不同主机的不同日志文件指向同一个日志流吗？

不支持配置多个日志源将数据发送到单个日志流。

代理调用哪些 API (或我应该将哪些操作添加到 IAM 策略中)？

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

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

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

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

在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`。

文档历史记录

下表介绍了对 Amazon CloudWatch Logs 用户指南的一些重要更改。

更改	描述	发行日期
Route 53 DNS 查询日志	您可以使用 CloudWatch Logs 存储有关 Route 53 收到的 DNS 查询的日志。有关更多信息，请参阅 Amazon Route 53 开发人员指南 中的什么是 Amazon CloudWatch Logs？ (p. 1)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 标记 Amazon CloudWatch Logs 中的日志组 (p. 33)。	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 (p. 49)。	2016 年 11 月 7 日
控制台可用性改进	改善了体验以更轻松地搜索、筛选和排查问题。例如，您现在可以筛选出某个日期和时间范围内的日志数据。有关更多信息，请参阅 查看发送到 CloudWatch Logs 的日志数据 (p. 30)。	2016 年 8 月 29 日
增加了对 Amazon CloudWatch Logs 和新的 CloudWatch Logs 指标的 AWS CloudTrail 支持	增加了对 CloudWatch Logs 的 AWS CloudTrail 支持。有关更多信息，请参阅 在 AWS CloudTrail 中记录 Amazon CloudWatch Logs API 调用 (p. 87)。	2016 年 3 月 10 日
增加了对 CloudWatch Logs 导出到 Amazon S3 的支持	增加了将 CloudWatch Logs 数据导出到 Amazon S3 的支持。有关更多信息，请参阅 将日志数据导出至 Amazon S3 (p. 68)。	2015 年 12 月 7 日
在 Amazon CloudWatch Logs 中增加了对 AWS CloudTrail 记录的事件的支持	您可以在 CloudWatch 中创建警报并接收 CloudTrail 捕获的特定 API 活动的通知，然后使用通知执行问题排查。	2014 年 11 月 10 日
增加了对 Amazon CloudWatch Logs 的支持	您可以使用 Amazon CloudWatch Logs 来监控、存储和访问系统、应用程序以及来自 Amazon Elastic Compute Cloud (Amazon EC2) 实例或其他来源的自定义日志文件。然后可以使用 Amazon CloudWatch 控制台、AWS CLI 中的 CloudWatch Logs 命令或使用 CloudWatch Logs 软件开发工具包，从 CloudWatch Logs 检索关联的日志数据。有关更多信息，请参阅 什么是 Amazon CloudWatch Logs？ (p. 1)。	2014 年 7 月 10 日

AWS 词汇表

有关最新 AWS 术语，请参阅 AWS General Reference 中的 AWS 词汇表。