aws

用户指南

适用于 Microsoft Windows 的 Amazon Kinesis 代理

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

适用于 Microsoft Windows 的 Amazon Kinesis 代理: 用户指南

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

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

Table of Contents

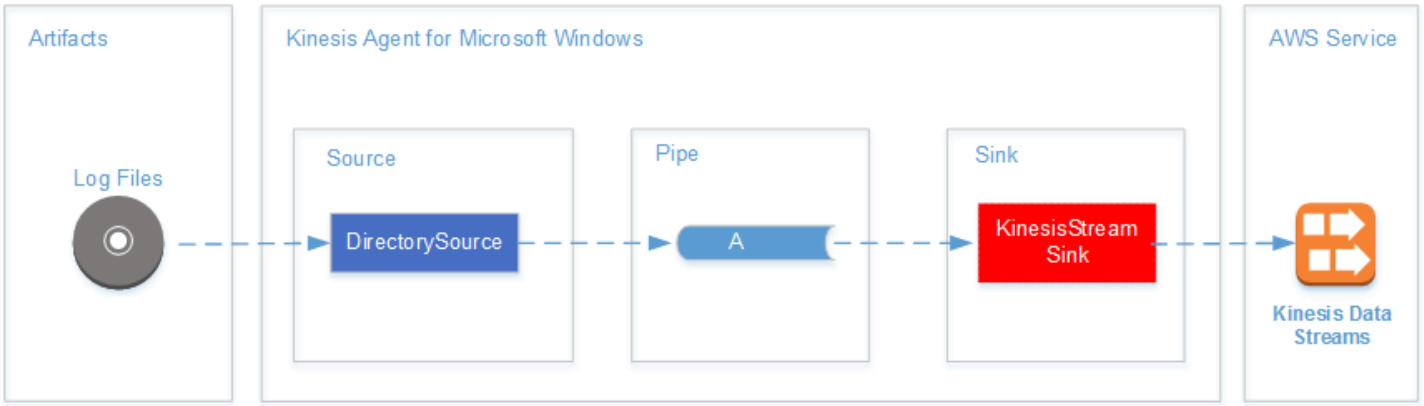
什么是 Amazon Kinesis Analytics？

Amazon Kinesis Analytics (Windows Kinesis Analytics) 是一种可配置和可扩展的代理。它在 Windows 桌面计算机和服务器队列上运行，可以在本地或 AWS 云中运行。Windows Kinesis Agent 高效可靠地收集、解析、转换日志、事件和指标，并流式传输到各种 AWS 服务，包括Kinesis Data Streams、Kinesis Data Firehose、Amazon CloudWatch, 和CloudWatch Logs (CloudWatch 日志)。

通过这些服务，您可以使用各种其他 AWS 服务存储、分析和可视化数据，包括以下服务：

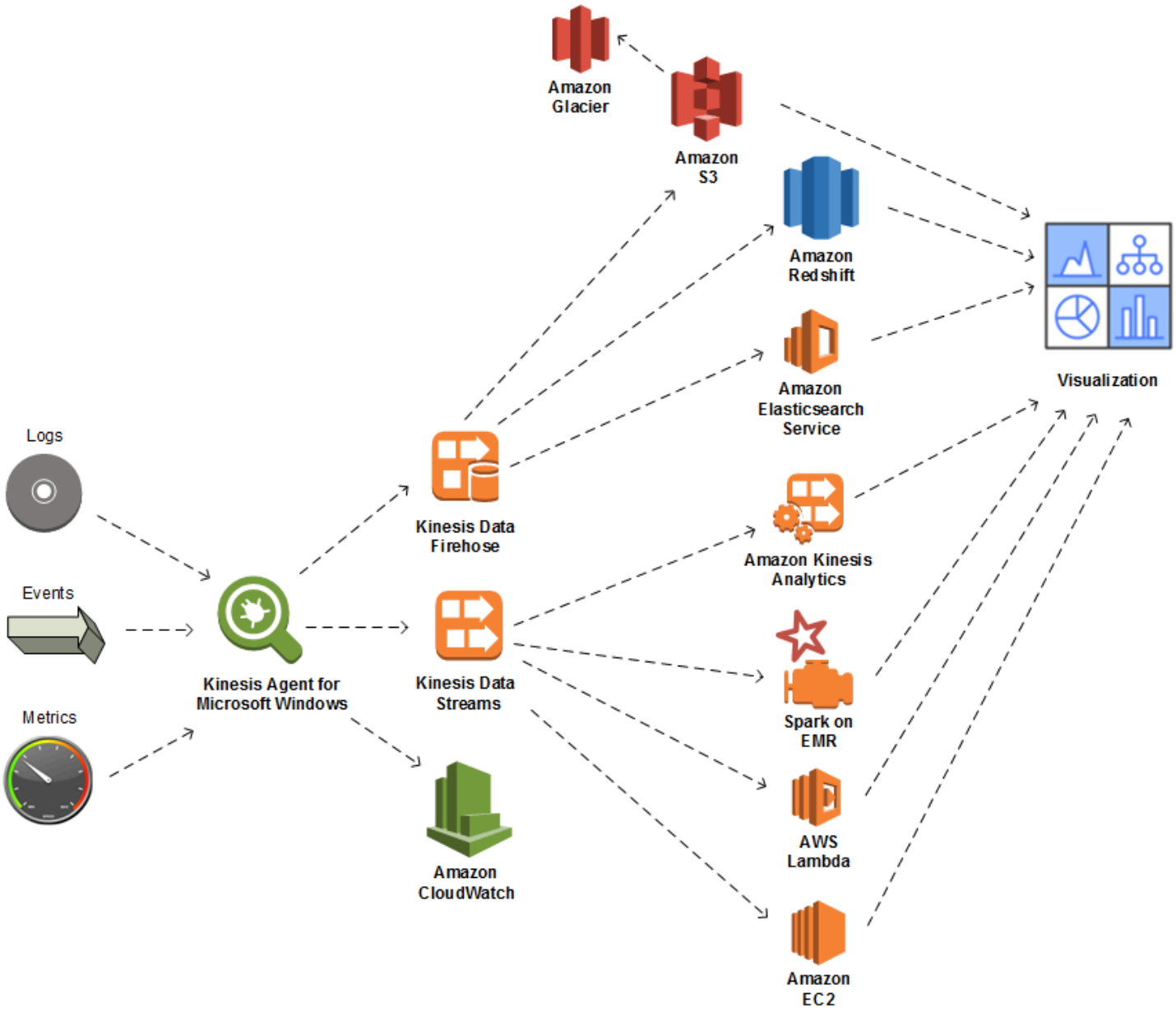
- Amazon Simple Storage Service (Amazon S3)
- Amazon Redshift
- Amazon Elasticsearch Service (Amazon ES)
- Kinesis Data Analytics
- Amazon QuickSight
- Amazon Athena
- Kibana

下图说明了 Windows Kinesis Agent 的简单配置，它将日志文件流式传输到 Kinesis Data Streams imple。



有关源、管道和接收器的更多信息，请参阅 面向微软 Windows 概念的 Amazon Kinesis 代理。

下图说明了使用流处理框架构建自定义实时数据管道的一些方法。这些框架包括 Kinesis Data Analytics、亚马逊 EMR 上的 Apache Spark 和 AWS Lambda。



主题

- 关于 AWS
- 有什么用途与 Windows Kinesis Analytics？
- Benefits
- 开始使用适用于 Windows 的 Kinesis 代理

关于 AWS

Amazon Web Services (AWS) 是数字基础设施服务的集合，您可在开发应用程序时使用这些服务。这些服务包括计算、存储、数据库、分析和应用程序同步 (消息发送和队列)。AWS 采用即付即用的服务模式。您只需为使用的服务或您的应用程序使用的服务付费。此外，AWS 还提供免费使用套餐，以便使其服务更易于进行原型设计和试验。在此套餐中，低于某种使用水平的服务是免费的。有关 AWS 成本和免费套餐，请参阅开始使用资源中心。要创建 AWS 账户，请打开 AWS 主页并注册。

有什么用途与 Windows Kinesis Analytics？

Windows Kinesis Analytics 具有以下特性和功能：

收集日志、事件和指标数据

Windows Kinesis Agent 收集、解析、转换服务器和台式机队列的日志、事件和指标，并将其流式传输到一个或多个 AWS 服务。这些服务接收的负载可以采用与原始源不同的格式。例如，日志可能以特定的文本格式（例如 syslog 格式）存储在服务器上。Windows Kinesis Agent 可以收集和解析该文本，并可选择将其转换为 JSON 格式，例如，在流式传输到 AWS 之前进行转换。这有助于通过一些使用 JSON 的 AWS 服务进行更简单的处理。流式传输到 Kinesis Data Analytics 可以连续处理流式传输到 Kinesis Data Analytics，以生成其他指标和汇总指标，从而为实时控制面板提供支持。您可以使用各种 AWS 服务（例如 Amazon S3）存储数据，具体取决于数据在数据管道中的下游使用方式。

与 AWS 服务集成

您可以将 Windows Kinesis Agent 配置为将日志文件、事件和指标发送到多个不同的 AWS 服务：

- Kinesis Data Firehose— 轻松将流式传输数据存储在 Amazon S3、Amazon Redshift、Amazon ES 或Splunk以便进一步分析。
- Kinesis Data Streams— 使用 Kinesis Data Analytics 或Amazon EMR。或者使用运行在Amazon EC2实例或自定义无服务器函数在AWS Lambda。

- CloudWatch— 在图形中查看流式处理指标，您可以将其组合到控制面板中。然后，设置由超出预设阈值的指标值触发的 CloudWatch 警报。
- CloudWatch Logs (CloudWatch 日志)— 存储流式处理日志和事件，并在 AWS 管理控制台中查看和搜索它们，或者在数据管道中进一步下游处理。

快速安装和配置

您只需几个步骤即可安装和配置 Windows Kinesis Agent。有关更多信息，请参阅 安装适用于 Windows 的 Kinesis 代理 和 为微软 Windows 配置 Amazon Kinesis 运行代理。一个简单的声明性配置文件指定以下内容：

- 要收集的日志、事件和指标的来源和格式。
- 要应用于收集的数据的转换。可以包括附加数据，并且可以转换和筛选现有数据。
- 流式传输最终数据的目标，以及流式处理负载的缓冲、分片和格式。

Windows Kinesis Agent 为常见的 Microsoft 企业服务生成的日志文件提供内置解析程序，例如：

- Microsoft Exchange
- SharePoint
- Active Directory 域控制器
- DHCP 服务器

无需持续管理

Windows Kinesis Agent 自动适应各种情况而不会丢失任何数据。其中包括日志轮换、重启后恢复以及临时的网络或服务中断。您可以将 Windows Kinesis Agent 配置为自动更新到新版本。在上述任何情况下都不需要操作员干预。

使用开放架构进行扩展

如果声明性功能和内置插件不足以监控服务器或台式机系统，则可以通过创建插件来扩展 Windows Kinesis Agent。新插件为日志、事件和指标启用新的源和目标。适用于 Windows 的 Kinesis 代理程序的源代码可以在https://github.com/awslabs/kinesis-agent-windows。

Benefits

Windows Kinesis Agent 为日志、事件和数据管道的指标执行初始数据收集、转换和流式处理。构建这些数据管道有许多优势：

分析和可视化

Windows Kinesis Agent 与 Kinesis Data Firehose 及其转换功能使其易于与多种不同的分析和可视化服务集成：

- Amazon QuickSight— 基于云的 BI 服务，可以从多个不同的源获取。Windows Kinesis 代理程序可以转换数据，并通过运动数 Kinesis Data Firehose 将其流式传输到 Amazon S3 和 Amazon Redshift。此过程可以使用 Amazon QuickSight 可视化从数据中发现深入的见解。
- Athena— 启用基于 SQL 的数据查询的交互式查询服务。适用于 Windows 的 Kinesis 代理程序可以将数据转换和流式传输到 Amazon S3 通过 Kinesis Data Firehose。然后，Athena 可以交互式地对该数据执行 SQL 查询，以快速检查和分析日志和事件。
- Kibana— 开源数据可视化工具。适用于 Windows 的 Kinesis 代理程序可以转换和流式传输数据到亚马逊 ES 通过 Kinesis Data Firehose。然后，您可以使用 Kibana 研究这些数据。创建和打开不同的可视化图形，包括直方图、折线图、饼图、热图和地理空间图形。

Security

包含 Kinesis Agent for Windows 的日志和事件数据分析管道可以检测和警报组织中的安全漏洞，从而帮助您阻止或停止攻击。

应用程序性能

Windows Kinesis Agent 可以收集有关应用程序或服务性能的日志、事件和指标数据。然后，完整数据管道可以分析这些数据。此分析通过检测和报告可能不明显的缺陷，帮助您提高应用程序和服务性能以及可靠性。例如，您可以检测服务 API 调用的执行时间的重大更改。与部署相关联时，此功能可帮助您找到并解决您拥有的服务的新性能问题。

服务操作

数据管道可以分析收集的数据，以预测潜在的操作问题，并提供如何避免服务中断的见解。例如，您可以分析日志、事件和指标，以确定当前和预计的容量使用情况，以便在服务用户受到影响之前在线提供额外容量。如果发生服务中断，您可以分析数据以确定在中断期间对客户的影响。

Auditing

数据管道可以处理 Windows Kinesis Agent 收集和转换的日志、事件和指标。然后，您可以使用各种 AWS 服务审核此处理的数据。例如，Kinesis Data Firehose 可以从 Windows Kinesis Agent 接收数据流，后者将数据存储在 Amazon S3 中。然后，您可以通过使用 Athena 执行交互式 SQL 查询来审核此数据。

Archiving

通常，最重要的操作数据是最近收集的数据。但是，对多年来收集的有关应用程序和服务的数据进行分析也很有用，例如，用于长期规划。保存大量数据可能成本高昂。适用于 Windows 的 Kinesis 代理可以通过 Kinesis 数据消防软管收集、转换和存储在 Amazon S3 中的数据。因此,Amazon S3 Glacier可用于降低存档旧数据的成本。

Alerting

适用于 Windows 的 Kinesis 代理程序将指标流式传输到 CloudWatch。反过来，您可以创建 CloudWatch 警报，以通过Amazon Simple Notification Service (Amazon SNS)当指标始终违反特定阈值时。这使工程师能够更好地了解其应用程序和服务的操作问题。

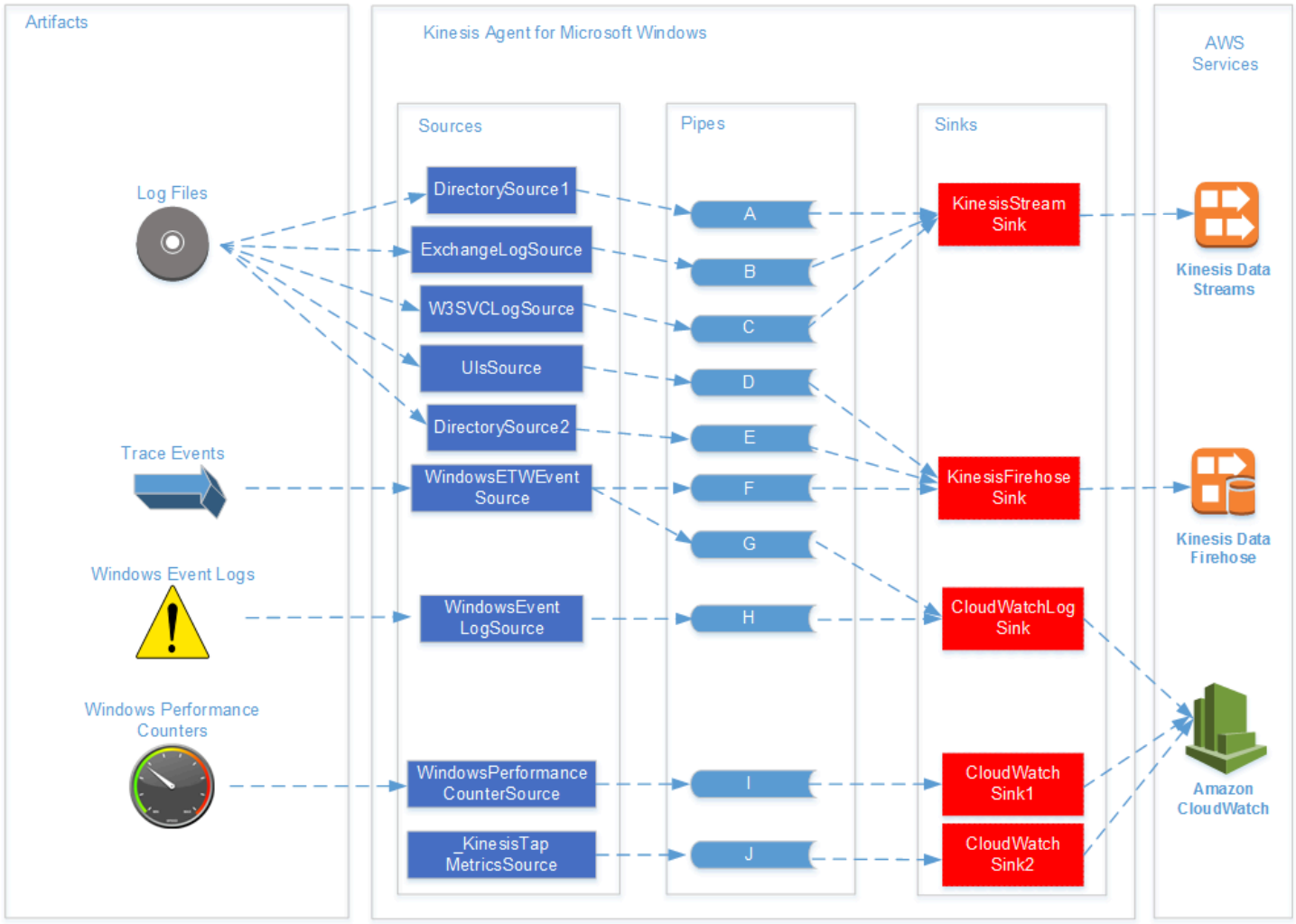
开始使用适用于 Windows 的 Kinesis 代理

要了解有关 Windows Kinesis Agent 的更多信息，我们建议您从以下部分入手：

- 面向微软 Windows 概念的 Amazon Kinesis 代理
- 适用于 Microsoft Windows 的 Amazon Kinesis 代理入门

面向微软 Windows 概念的 Amazon Kinesis 代理

了解面向微软 Windows 的 Amazon Kinesis 代理（适用于 Windows 的 Kinesis 代理）的主要概念可让您更轻松地在台式机和服务器机群上收集并流式传输数据到数据管道的剩余部分，以进行处理。



此数据管道图说明了下列组件和流程：

服务器和台式机具有日志文件、事件和指标等构件，这些构件由 Windows 的一个或多个 Kinesis 代理收集来源。例如，可以选择从纯文本文件格式传输数据到对象。

然后，数据（在对象中或者文本形式）可以流向 Windows 的一个或多个 Kinesis 代理管道。一个管道将一个源连接到 Windows 的一个 Kinesis 代理sink。管道可以选择性地筛选出不必要的数据。

接收器可以选择性的将解析的数据转换为 JSON 或 XML 格式的对象。接收器将数据发送到特定的 AWS 服务，例如 Kinesis Data Streams、Kinesis Data Firehose 或 Amazon CloudWatch。

使用多个管道，一个源可以将相同的数据发送到多个接收器（例如，参阅图中的管道 F 和 G）。使用多个管道，不同源可以将数据流式传输到单个接收器（例如，参阅图中的管道 A、B 和 C）。还可以使用多个管道将数据从多个接收器流式传输到多个源。源、接收器和管道具有类型，同一个类型中可以有多个源、接收器或管道。

有关声明源、接收器和管道的配置文件示例，请参阅适用于 Windows 的 Kinesis 代理配置示例。

主题

数据管道

A数据管道用于收集、处理、可视化以及可能生成应用程序和服务的警报。适用于 Windows 的 Kinesis Agent 可以在开始时融入数据管道，其中日志、事件和衡量指标是从台式计算机或服务器的队列中收集的。Windows Kinesis 代理将收集的数据流式传输到构成数据管道剩余部分的各种 AWS 服务。数据管道具有目的，例如实时可视化特定服务的运行状况数据，用于帮助工程师更高效地操作该服务。服务运行状况数据管道可以执行以下任一操作：

- 在问题影响到服务客户的体验之前，向工程师发出问题警报。
- 通过展示资源使用趋势，帮助工程师高效地管理服务的成本。这些趋势使得他们可以相应调整资源级别，甚至实施自动扩展场景。
- 针对服务客户报告的问题，提供问题根本原因的见解。这可以加快解决这些问题并减少支持成本。

有关使用适用于 Windows 的 Kinesis 代理构造数据管道的分步示例，请参阅教程：使用适用于 Windows 的 Kinesis 代理将 JSON 日志文件流式传输到 Amazon S3。

Sources

适用于 Windows 的 Kinesis 代理source收集日志、事件或指标。源从特定数据类型的特定创建器，根据源的类型收集该数据。例如，`DirectorySource` 类型从文件系统中的特定目录收集日志文件。如果数据尚未结构化（如某些类型的日志文件），则在将文本表示解析为某些结构化格式时，源会非常有用。每个源对应于一个特定的源声明在适用于 Windows 的 Kinesis 代理`appsettings.json`配置文

件。源声明提供必要详细信息，用于配置源以根据特定数据收集需求来定制源。可以配置的详细信息类型因源类型而异。例如，`DirectorySource` 源类型需要存放日志文件的目录的规范。

有关源类型和源声明的详细信息，请参阅源声明。

Sinks

适用于 Windows 的 Kinesis 代理sink获取由 Windows 源 Kinesis 代理收集的数据，并将该数据流式传输到构成数据管道剩余部分的多种可能 AWS 服务之一。每个水槽对应于一个特定的接收器声明在适用于 Windows 的 Kinesis 代理`appsettings.json`配置文件。接收器声明提供必要详细信息，用于配置接收器以根据特定数据流式传输需求来定制接收器。可以配置的详细信息类型因接收器类型而异。例如，一些接收器类型允许接收器声明针对向其提供的数据指定特定序列化 `Format`。在接收器声明中指定了此选项时，首先对收集的数据进行序列化，然后再将数据流式传输到与接收器关联的 AWS 服务。

有关接收器类型和接收器声明的更多信息，请参阅接收器声明。

Pipes

适用于 Windows 的 Kinesis 代理管道将 Windows 源的 Kinesis 代理的输出连接到 Windows 接收器的 Kinesis 代理的输入。在数据流经管道时，可以选择对数据进行转换。每个管道对应于 Windows 的 Kinesis 代理中的一个特定管道声明`appsettings.json`配置文件。管道声明提供配置管道的必要详细信息，例如管道的源和接收器。

有关管道类型和管道声明的更多信息，请参阅管道声明。

适用于 Microsoft Windows 的 Amazon Kinesis 代理入门

您可以使用适用于微软 Windows 的 Amazon Kinesis 代理程序（适用于 Windows 的 Kinesis 代理）从 Windows 队列收集、分析、转换和流式传输日志、事件和指标到各种 AWS 服务。以下信息包含用于 Windows 的 Kinesis 代理程序的先决条件和分步说明。

主题

Prerequisites

在安装 Windows Kinesis 代理之前，请确保您具备以下先决条件：

- 熟悉 Windows 的 Kinesis 代理概念。有关更多信息，请参阅 面向微软 Windows 概念的 Amazon Kinesis 代理。
- AWS 账户，用于使用与数据管道相关的各种 AWS 服务。有关创建和配置 AWS 账户的更多信息，请参阅设置 AWS 账户。
- Microsoft .NET Framework 4.6 或更高版本，安装在每个将运行 Windows 代理程序的台式机或服务器上。有关更多信息，请参阅 Microsoft .NET 文档中的安装面向开发人员的 .NET Framework。

  要确定台式机或服务器上安装的 .NET Framework 的最新版本，请使用以下 PowerShell 脚本：

  ```
  [System.Version](
  (Get-ChildItem 'HKLM:\SOFTWARE\Microsoft\NET Framework Setup\NDP' -recurse `
  | Get-ItemProperty -Name Version -ErrorAction SilentlyContinue `
  | Where-Object { ($_.PSChildName -match 'Full') } `
  | Select-Object Version | Sort-Object -Property Version -Descending)[0]).Version
  ```

- 您希望从适用于 Windows 的 Kinesis 代理发送数据的流（如果使用 Amazon Kinesis Data Streams）。使用Kinesis Data Streams 控制台，AWS CLI，或者适用于 Windows PowerShell 的 AWS 工具。有关更多信息，请参阅 。创建和更新数据流中的Amazon Kinesis Data Streams 开发人员指南。

- 您希望从 Windows Kinesis 代理发送数据的 Firehose 传输流（如果使用 Amazon Kinesis Data Firehose）。创建交付流使用Kinesis Data Firehose 控制台，AWS CLI，或者适用于 Windows PowerShell 的 AWS 工具。有关更多信息，请参阅 。创建 Amazon Kinesis Data Firehose 传输流中的Amazon Kinesis Data Firehose 开发人员指南。

设置 AWS 账户

如果您还没有 AWS 账户，请完成以下步骤创建一个账户。

如何注册 AWS 账户

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

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

自行创建管理员用户并将该用户添加到管理员组（控制台）

1. 登录到IAM 控制台作为帐户所有者，方法是选择根用户并输入您的 AWS 账户电子邮件地址。在下一页上，输入您的密码。

   > **Note**
   >
   > 强烈建议您遵守以下最佳实践：使用**Administrator**IAM 用户，遵守并妥善保存根用户凭证。只在执行少数账户和服务管理任务时才作为根用户登录。

2. 在导航窗格中，选择用户，然后选择添加用户。
3. 对于 User name (用户名)，输入 **Administrator**。
4. 选中 AWS Management Console access (AWS 管理控制台访问) 旁边的复选框。然后选择自定义密码，并在文本框中输入新密码。
5. （可选）默认情况下，AWS 要求新用户在首次登录时创建新密码。您可以清除 User must create a new password at next sign-in (用户必须在下次登录时创建新密码) 旁边的复选框以允许新用户在登录后重置其密码。
6. 选择后续：权限。
7. 在设置权限下，选择将用户添加到组。
8. 选择创建组。

9. 在 Create group (创建组) 对话框中，对于 Group name (组名称)，输入 **Administrators**。
10. 选择筛选策略，然后选择AWS 托管-工作职能来过滤表格内容。
11. 在策略列表中，选中 AdministratorAccess 的复选框。然后选择 Create group (创建组)。

> **Note**
>
> 您必须先激活 IAM 用户和角色对账单的访问权限，然后才能使用 `AdministratorAccess` 权限访问 AWS 账单和成本管理控制台。为此，请按照"向账单控制台委派访问权限"教程第 1 步中的说明进行操作。

12. 返回到组列表中，选中您的新组所对应的复选框。如有必要，选择 Refresh 以在列表中查看该组。
13. 选择后续：标签。
14. （可选）通过以键值对的形式附加标签来向用户添加元数据。有关在 IAM 中使用标签的更多信息，请参阅标记 IAM 实体中的IAM 用户指南。
15. 选择后续：审核以查看要添加到新用户的组成员资格的列表。如果您已准备好继续，请选择 Create user。

您可使用此相同的流程创建更多的组和用户，并允许您的用户访问 AWS 账户资源。要了解有关使用策略限制用户对特定 AWS 资源的权限的信息，请参阅访问管理和示例策略。

注册 AWS 并创建管理员账户

1. 如果您没有 AWS 账户，请打开https://aws.amazon.com/。选择创建 AWS 账户，然后按照联机说明操作。

   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。
2. 登录 AWS 管理控制台，并通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
3. 在导航窗格中，选择 Groups (组)，然后选择 Create New Group (创建新组)。
4. 对于组名，输入组的名称，例如 **Administrators**，然后选择 下一步。
5. 在策略列表中，选中 AdministratorAccess 策略旁边的复选框。您可以使用 Filter (筛选) 菜单和 Search (搜索) 框来筛选策略列表。
6. 选择 Next Step。选择创建组，您的新组将显示在组名下。
7. 在导航窗格中选择用户，然后选择创建新用户。

8. 在框 1 中，输入用户名，清除为每个用户生成访问密钥旁的复选框，然后选择创建。
9. 在用户列表中，选择您刚创建的用户的名称（而不是复选框）。您可以使用搜索框来搜索该用户名称。
10. 选择组选项卡，然后选择将用户添加到多个组。
11. 选中管理员组旁的复选框，然后选择添加到多个组。
12. 选择 Security Credentials (安全证书) 选项卡。在 Sign-In Credentials (登录证书) 下，选择 Manage Password (管理密码)。
13. 选择分配自定义密码，在密码和确认密码框中输入密码，然后选择应用。

安装适用于 Windows 的 Kinesis 代理

在 Windows 上有三种方法来安装适用于 Windows 的 Kinesis 代理：

- 使用 MSI（Windows 安装程序包）进行安装。
- 从安装AWS Systems Manager，这是一组用于管理服务器和台式机的服务。
- 运行 PowerShell 脚本。

> **Note**
>
> 下列说明偶尔会使用术语 KinesisTap 和 AWSKinesisTap。这些单词表达的含义与 Windows Kinesis 代理相同，但在执行这些指令时，您必须原样指定它们。

使用 MSI 安装适用于 Windows 的 Kinesis 代理

您可以从 Windows MSI 软件包下载最新的 Kinesis 代理程序包从GitHub 上的动态代理窗口存储库。下载 MSI 后，请使用 Windows 启动它并按照安装程序提示进行操作。安装后，你可以像卸载任何 Windows 应用程序一样。

此外，也可以使用密西西克命令以静默方式安装、打开日志记录和卸载，如以下示例所示。Replace*AWSKinesisTap.1.1.216.4.msi* with the appropriate version of Kinesis Agent for Windows for your application.

要以静默方式安装适用于 Windows 的 Kinesis 代理：

```
msiexec /i AWSKinesisTap.1.1.216.4.msi /q
```

将安装消息记录到名为***logfile.log***：

```
msiexec /i AWSKinesisTap.1.1.216.4.msi /q /L*V logfile.log
```

要使用命令提示符卸载适用于 Windows 的 Kinesis 代理，请执行以下操作：

```
msiexec.exe /x {ADAB3982-68AA-4B45-AE09-7B9C03F3EBD3} /q
```

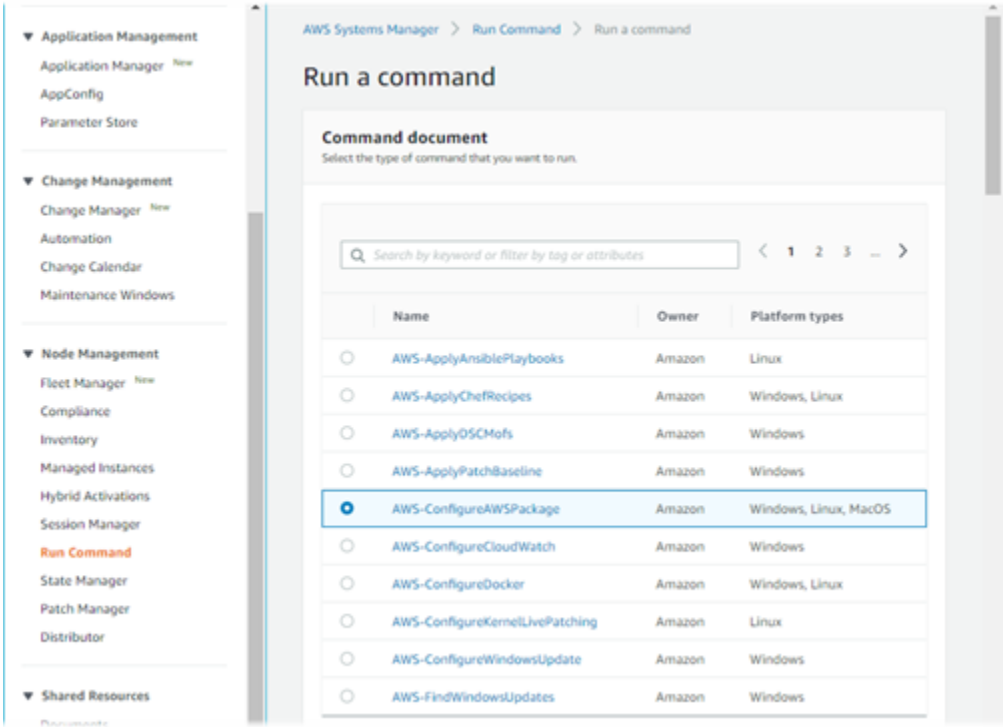
使用 AWS Systems Manager 安装适用于 Windows 的 Kinesis 代理

使用 Systems Manager Run Command，按照以下步骤安装适用于 Windows 的 Kinesis 代理。有关 Run Command 的更多信息，请参阅AWS Systems Manager Run Command中的AWS Systems Manager 用户指南。除了使用 Systems Manager Run Command 之外，您还可以使用 Systems Manager维护时段和状态管理器来自动部署适用于 Windows 的 Kinesis 代理程序。

> **Note**
>
> 适用于 Windows 的 Kinesis 代理的 Systems Manager 安装在AWS Systems Manager但以下情况除外：
>
> - cn-north-1
> - cn-northwest-1
> - 所有 AWS GovCloud 区域。

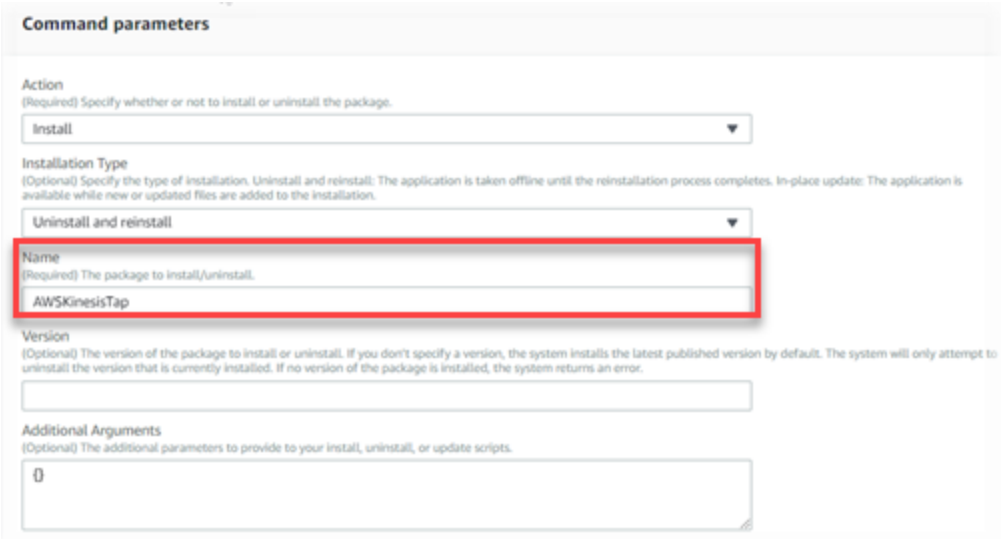
使用 Systems Manager 安装适用于 Windows 的 Kinesis 代理

1. 确保在您希望安装适用于 Windows 的 Kinesis 代理程序的实例上，已经安装了版本 2.2.58.0 或更高版本。有关更多信息，请参阅 。在 Windows 实例上安装和配置 SSM 代理中的AWS Systems Manager 用户指南。
2. 打开 AWS Systems Manager 控制台，网址为https://console.aws.amazon.com/systems-manager/。
3. 从导航窗格的Node Management中，选择Run Command，然后选择Run Command。
4. 从命令文档列表中，选择AWS 配置 AWS-ConfigreAWSPackage文档.



5. UDER命令参数, 用于名称，输入冬奥会。将其他设置保留默认值。

> **Note**
>
> 离开版本空白以指定 AWSKinesiStap 软件包的最新版本。（可选）您可以输入要安装的特定版本。



6. UDER目标中，指定要对其运行命令的实例。您可以选择根据与实例关联的标签指定实例，也可以手动选择实例，也可以指定包含实例的资源组。
7. 将所有其他设置保留默认值，然后选择运行。

使用 PowerShell 安装适用于 Windows 的 Kinesis 代理

使用文本编辑器将以下命令复制到文件中，然后将其保存为 PowerShell 脚本。我们使用`InstallKinesisAgent.ps1`，请参阅以下示例。

```
Param(
    [ValidateSet("prod", "beta", "test")]
    [string] $environment = 'prod',
    [string] $version,
    [string] $baseurl
)

# Self-elevate the script if required.
if (-Not ([Security.Principal.WindowsPrincipal]
 [Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltI
 'Administrator')) {
    if ([int](Get-CimInstance -Class Win32_OperatingSystem | Select-Object -
ExpandProperty BuildNumber) -ge 6000) {
        $CommandLine = '-File "' + $MyInvocation.MyCommand.Path + '" ' +
 $MyInvocation.UnboundArguments
        Start-Process -FilePath PowerShell.exe -Verb Runas -ArgumentList $CommandLine
        Exit
    }
}

# Allows input to change base url. Useful for testing.
if ($baseurl) {
    if (!$baseUrl.EndsWith("/")) {
        throw "Invalid baseurl param value. Must end with a trailing forward slash
 ('/')"
    }

    $kinesistapBaseUrl = $baseurl
} else {
    $kinesistapBaseUrl = "https://s3-us-west-2.amazonaws.com/kinesis-agent-windows/
downloads/"
}
```

```
Write-Host "Using $kinesistapBaseUrl as base url"

$webClient = New-Object System.Net.WebClient

try {
    $packageJson = $webClient.DownloadString($kinesistapBaseUrl + 'packages.json' + '?
_t=' + [System.DateTime]::Now.Ticks) | ConvertFrom-Json
} catch {
    throw "Downloading package list failed."
}


if ($version) {
    $kinesistapPackage = $packageJson.packages | Where-Object { $_.packageName -eq
 "AWSKinesisTap.$version.nupkg" }

    if ($null -eq $kinesistapPackage) {
        throw "No package found matching input version $version"
    }
} else {
    $packageJson = $packageJson.packages | Where-Object { $_.packageName -match
 ".nupkg" }
    $kinesistapPackage = $packageJson[0]
}

$packageName = $kinesistapPackage.packageName
$checksum = $kinesistapPackage.checksum

#Create %TEMP%/kinesistap if not exists
$kinesistapTempDir = Join-Path $env:TEMP 'kinesistap'
if (![System.IO.Directory]::Exists($kinesistapTempDir)) {[void]
[System.IO.Directory]::CreateDirectory($kinesistapTempDir)}

#Download KinesisTap.x.x.x.x.nupkg package
$kinesistapNupkgPath = Join-Path $kinesistapTempDir $packageName
$webClient.DownloadFile($kinesistapBaseUrl + $packageName, $kinesistapNupkgPath)
$kinesistapUnzipPath = $kinesistapNupkgPath.Replace('.nupkg', '')

# Calculates hash of downloaded file. Downlevel compatible using .Net hashing on PS < 4
if ($PSVersionTable.PSVersion.Major -ge 4) {
    $calculatedHash = Get-FileHash $kinesistapNupkgPath -Algorithm SHA256
    $hashAsString = $calculatedHash.Hash.ToLower()
} else {
    $sha256 = New-Object System.Security.Cryptography.SHA256CryptoServiceProvider
```

```
    $calculatedHash =
 [System.BitConverter]::ToString($sha256.ComputeHash([System.IO.File]::ReadAllBytes($kinesistap
    $hashAsString = $calculatedHash.Replace("-", "").ToLower()
}

if ($checksum -eq $hashAsString) {
 Write-Host 'Local file hash matches checksum.' -ForegroundColor Green
} else {
 throw ("Get-FileHash does not match! Package may be corrupted.")
}

#Delete Unzip path if not empty
if ([System.IO.Directory]::Exists($kinesistapUnzipPath)) {Remove-Item -Path
 $kinesistapUnzipPath -Recurse -Force}

#Unzip KinesisTap.x.x.x.x.nupkg package
$null =
 [System.Reflection.Assembly]::LoadWithPartialName('System.IO.Compression.FileSystem')
[System.IO.Compression.ZipFile]::ExtractToDirectory($kinesistapNupkgPath,
 $kinesistapUnzipPath)

#Execute chocolaeyInstall.ps1 in the package and wait for completion.
$installScript = Join-Path $kinesistapUnzipPath '\tools\chocolateyInstall.ps1'
& $installScript

# Verify service installed.
$serviceName = 'AWSKinesisTap'
$service = Get-Service -Name $serviceName -ErrorAction Ignore
if ($null -eq $service) {
    throw ("Service not installed correctly.")
} else {
    Write-Host "Kinesis Tap Installed." -ForegroundColor Green
    Write-Host "After configuring run the following to start the service: Start-Service
 -Name $serviceName." -ForegroundColor Green
}
```

打开提升的命令提示符窗口。在文件下载到的目录中，使用以下命令来运行脚本：

```
PowerShell.exe -File ".\InstallKinesisAgent.ps1"
```

若要安装适用于 Windows 的特定版本的 Kinesis 代理，请将`-version`选项：

```
PowerShell.exe -File ".\InstallKinesisAgent.ps1" -version "version"
```

Replace*version*使用 Windows 版本号的有效 Kinesis 代理程序。有关版本信息，请参阅GitHub 上的动态代理窗口存储库。

有许多部署工具可以远程执行 PowerShell 脚本。它们可用于在服务器机群或台式机上自动完成的 Windows Kinesis 代理安装。

配置和启动适用于 Windows 的 Kinesis 代理

安装适用于 Windows 的 Kinesis 代理后，您必须配置并启动代理。在此之后，应该无需进一步的操作干预。

配置和启动适用于 Windows 的 Kinesis 代理的步骤

1. 创建和部署适用于 Windows 配置文件的 Kinesis 代理。此文件配置源、接收器和管道，以及其他全局配置项。

   有关 Windows 配置 Kinesis 代理的更多信息，请参阅为微软 Windows 配置 Amazon Kinesis 运行代理。

   有关您可以自定义并安装的完整配置文件示例，请参阅适用于 Windows 的 Kinesis 代理配置示例。
2. 打开一个提升的 PowerShell 命令提示符窗口，然后使用以下 PowerShell 命令启动 Windows Kinesis 代理程序：

   ```
   Start-Service -Name AWSKinesisTap
   ```

为微软 Windows 配置 Amazon Kinesis 运行代理

在启动适用于微软 Windows 的 Amazon Kinesis 代理之前，您必须创建配置文件并部署它。该配置文件提供了所需的信息，用于将 Windows 服务器和台式计算机上的数据收集、转换并流式传输到不同的 AWS 服务。配置文件定义一组源、接收器和将源连接到接收器的管道，以及可选的转换。

Windows 配置文件的 Kinesis 代理名为`appsettings.json`。将此文件部署到 `%PROGRAMFILES%\Amazon\AWSKinesisTap`。

主题

基本配置结构

Amazon Kinesis 的 Windows 配置文件配置文件的基本结构是具有以下模板的 JSON 文档：

```
{
     "Sources": [ ],
     "Sinks": [ ],
     "Pipes": [ ]
}
```

- `Sources` 的值是一个或多个源声明。
- `Sinks` 的值是一个或多个接收器声明。
- `Pipes` 的值是一个或多个管道声明。

有关的 Windows 源、接收器和管道概念，请参阅面向微软 Windows 概念的 Amazon Kinesis 代理。

以下示例是一个完整的appsettings.json配置文件，其中配置以将 Windows 应用程序日志事件流式传输到 Kinesis 数据消防器。

```
{
  "Sources": [
    {
      "LogName": "Application",
      "Id": "ApplicationLog",
      "SourceType": "WindowsEventLogSource"
    }
  ],
  "Sinks": [
    {
      "StreamName": "ApplicationLogFirehoseStream",
      "Region": "us-west-2",
      "Id": "MyKinesisFirehoseSink",
      "SinkType": "KinesisFirehose"
    }
  ],
  "Pipes": [
    {
      "Id": "ApplicationLogTotestKinesisFirehoseSink",
      "SourceRef": "ApplicationLog",
      "SinkRef": "MyKinesisFirehoseSink"
    }
  ]
}
```

有关各种类型声明的信息，请参阅以下章节：

- 源声明
- 接收器声明
- 管道声明

配置区分大小写

JSON 格式的文件通常区分大小写，并且您也应假定 Windows 配置文件中 Kinesis 所有键和值均区分大小写。appsettings.json 配置文件中的某些键和值不区分大小写，例如：

- 接收器的 Format 键/值对的值。有关更多信息，请参阅 接收器声明。

- 源的 SourceType 键/值对的值、接收器的 SinkType 键/值对的值以及管道和插件的 Type 键/值对的值。
- RecordParser 源的 DirectorySource 键/值对的值。有关更多信息，请参阅 DirectorySource 配置。
- 源的 InitialPosition 键/值对的值。有关更多信息，请参阅 书签配置。
- 变量替换的前缀。有关更多信息，请参阅 配置接收器变量替换。

源声明

在适用于微软 Windows 的 Amazon Kinesis 代理程序中，源声明描述应该在何处收集哪些日志、事件和指标数据。它们还可以选择性地指定用于解析数据的信息，以便对数据进行转换。以下部分介绍了针对 Windows 的 Kinesis 代理中可用的内置源类型的配置。由于 Windows 的 Kinesis 代理是可扩展的，因此，您可以添加自定义源类型。每种源类型通常都需要配置对象中与该源类型相关的特定键/值对。

所有源声明都必须至少包含以下键/值对：

Id

一个唯一字符串，用于标识配置文件中的特定源对象。

SourceType

此源对象的源类型的名称。源类型指定此源对象正在收集的日志、事件或指标数据的来源。它还控制可声明源的其他方面。

有关使用不同类型的源声明的完整配置文件的示例，请参阅从不同源流式传输到 Kinesis Data Streams。

主题

- DirectorySource 配置
- ExchangeLogSource 配置
- W3SVCLogSource 配置
- UlsSource 配置
- WindowsEventLogSource 配置
- 窗口七个日志轮询源配置
- WindowsETWEventSource 配置

DirectorySource 配置

Overview

DirectorySource 源类型从存储在指定目录中的文件收集日志。由于日志文件有许多不同的格式，因此，DirectorySource 声明允许您指定日志文件中数据的格式。然后，您可以将日志内容转换为标准格式（如 JSON 或 XML），然后再将该内容流式传输到各种 AWS 服务。

以下是示例 DirectorySource 声明：

```
{
    "Id": "myLog",
    "SourceType": "DirectorySource",
    "Directory": "C:\\Program Data\\MyCompany\\MyService\\logs",
    "FileNameFilter": "*.log",
    "IncludeSubdirectories": true,
    "IncludeDirectoryFilter": "cpu\\cpu-1;cpu\\cpu-2;load;memory",
    "RecordParser": "Timestamp",
    "TimestampFormat": "yyyy-MM-dd HH:mm:ss.ffff",
    "Pattern": "\\d{4}-\\d{2}-\\d(2}",
    "ExtractionPattern": "",
    "TimeZoneKind": "UTC",
    "SkipLines": 0,
    "Encoding": "utf-16",
    "ExtractionRegexOptions": "Multiline"
}
```

所有 DirectorySource 声明都可以提供以下键/值对：

SourceType

必须是文本字符串 "DirectorySource"（必需）。

Directory

包含日志文件的目录的路径（必需）。

`FileNameFilter`

（可选）根据通配符文件命名模式限制从中收集日志数据的目录中的文件集。如果您有多个日志文件名模式，则此功能允许您使用单个`DirectorySource`，如以下示例所示。

```
FileNameFilter: "*.log|*.txt"
```

系统管理员有时会在归档日志文件之前压缩日志文件。如果您指定`"*.*"`在`FileNameFilter`，已知的压缩文件现在将被排除。此功能可防止`.zip`、`.gz`, 和`.bz2`文件被意外流式传输。如果未指定此键/值对，则默认情况下会收集目录中所有文件中的数据。

`IncludeSubdirectories`

指定将子目录监视到受操作系统限制的任意深度。此功能对于监控具有多个网站的 Web 服务器非常有用。您也可以使用`IncludeDirectoryFilter`属性以仅监视过滤器中指定的某些子目录。

`RecordParser`

指定 `DirectorySource` 源类型应如何解析在指定目录中找到的日志文件。此键/值对是必需的，有效值如下所示：

- `SingleLine`— 日志文件的每一行都是一条日志记录。
- `SingleLineJson`— 日志文件的每一行都是一条 JSON 格式的日志记录。当您想使用对象修饰向 JSON 添加其他键/值对时，此解析程序非常有用。有关更多信息，请参阅 配置接收器修饰。有关使用 `SingleLineJson` 记录解析程序的示例，请参阅教程：使用适用于 Windows 的 Kinesis 代理将 JSON 日志文件流式传输到 Amazon S3。
- `Timestamp`— 一行或多行可以包含一条日志记录。日志记录的开头为时间戳。此选项要求指定 `TimestampFormat` 键/值对。
- `Regex`— 每条记录都以匹配特定正则表达式的文本开头。此选项要求指定 `Pattern` 键/值对。
- `SysLog`— 指示日志文件写入syslog标准格式。根据该规范将日志文件解析为记录。
- `Delimited`— 一个更简单的 Regex 记录解析程序版本，其中日志记录中的数据项由一致的分隔符分隔。此选项比 Regex 解析程序更容易使用且执行速度更快，当此选项可用时，它是首选。在使用此选项时，您必须指定 `Delimiter` 键/值对。

`TimestampField`

指定哪个 JSON 字段包含记录的时间戳。它仅可与 `SingleLineJson` `RecordParser` 结合使用。键/值对是可选的。如果未指定此项，Windows 的 Kinesis 代理使用读取记录的时间作为时间戳。指定此键/值对的一个优点是，由 Windows 的 Kinesis 代理生成的延迟统计信息更准确。

`TimestampFormat`

指定如何解析与记录关联的日期和时间。该值可以是字符串 epoch 或 .NET 日期/时间格式字符串。如果值为 epoch，则根据 UNIX 纪元时间来解析时间。有关 UNIX 纪元时间的更多信息，请参阅 Unix 时间。有关 .NET 日期/时间格式字符串的更多信息，请参阅 Microsoft .NET 文档中的自定义日期和时间格式字符串。此键/值对仅在以下情况下是必需的：指定了 `Timestamp` 记录解析程序或将 `SingleLineJson` 记录解析程序与 `TimestampField` 键/值对一起指定。

`Pattern`

指定必须与可能的多行记录的第一行匹配的正则表达式。此键/值对仅对于 `Regex` 记录解析程序是必需的。

`ExtractionPattern`

指定应使用命名组的正则表达式。使用此正则表达式解析记录，并且命名组构成已解析记录的字段。然后，这些字段将用作构建 JSON 或 XML 对象或文档的基础，这些对象或文档随后由接收器流式传输到各种 AWS 服务。键/值对是可选的，并且可用于`Regex`记录解析器和时间戳解析器。

`Timestamp` 组名称经过特殊处理，因为它向 `Regex` 解析程序指示哪个字段包含每个日志文件中每条记录的日期和时间。

`Delimiter`

指定分隔每条日志记录中每个项目的字符或字符串。此键/值对必须（且只能）用于 `Delimited` 记录解析程序。使用双字符序列 `\t` 表示制表符。

`HeaderPattern`

指定用于匹配日志文件中包含记录标头集的行的正则表达式。如果日志文件不包含任何标头信息，请使用 `Headers` 键/值对指定隐式标题。`HeaderPattern` 键/值对是可选的，并且仅对 `Delimited` 记录解析程序有效。

> **Note**
>
> 列的空（0 长度）标头条目导致从 `DirectorySource` 解析输出的最终输出中筛选出该列的数据。

`Headers`

指定使用指定分隔符解析的数据列的名称。此键/值对是可选的，并且仅对 `Delimited` 记录解析程序有效。

> **Note**
>
> 列的空（0 长度）标头条目导致从 DirectorySource 解析输出的最终输出中筛选出该列的数据。

`RecordPattern`

指定标识日志文件中包含记录数据的行的正则表达式。除了 HeaderPattern 标识的可选标题行之外，在记录处理期间将忽略与指定的 RecordPattern 不匹配的行。此键/值对是可选的，并且仅对 Delimited 记录解析程序有效。如果未提供此项，则默认情况下将任何与可选 HeaderPattern 或可选 CommentPattern 不匹配的行视为包含可解析记录数据的行。

`CommentPattern`

指定一个正则表达式，该表达式标识在解析日志文件中的数据之前应排除的日志文件中的行。此键/值对是可选的，并且仅对 Delimited 记录解析程序有效。如果未提供此项，则默认情况下将任何与可选 HeaderPattern 不匹配的行视为包含可解析记录数据的行，除非指定了 RecordPattern。

`TimeZoneKind`

指定是应在本地时区还是 UTC 时区中考虑日志文件中的时间戳。这是可选的，默认值为 UTC。此键/值对的唯一有效值为 Local 或 UTC。如果未指定 TimeZoneKind 或值为 UTC，则绝不会更改时间戳。时间戳将转换为 UTC 时TimeZoneKind值为Local，接收时间戳的接收器为 CloudWatch Logs，或将解析后的记录发送到其他接收器。不会转换消息中嵌入的日期和时间。

`SkipLines`

在指定此项时，控制在记录解析发生之前在每个日志文件的开头忽略的行数。这是可选的，默认值为 0。

编码

默认情况下，Windows 的 Kinesis 代理可以自动检测从字节标记的编码。但是，自动编码可能无法在某些较旧的 unicode 格式上正常工作。以下示例指定流式传输 Microsoft SQL Server 日志所需的编码。

```
"Encoding": "utf-16"
```

有关编码名称的列表，请参阅编码列表在微软 .NET 文档中。

提取正则表达式选项

您可以使用ExtractionRegexOptions来简化正则表达式。键/值对是可选的。默认为 "None"。

以下示例指定"."表达式匹配任何字符，包括\r\n。

```
"ExtractionRegexOptions" = "Multiline"
```

有关提取正则表达式选项的可能字段的列表，请参阅正则表达式选项枚举在微软 .NET 文档中。

Regex 记录解析程序

您可以将 Regex 记录解析程序与 TimestampFormat、Pattern 和 ExtractionPattern 键/值对结合使用来解析非结构化文本日志。例如，假设您的日志文件如下所示：

```
[FATAL][2017/05/03 21:31:00.534][0x00003ca8][0000059c][][ActivationSubSystem]
[GetActivationForSystemID][0] 'ActivationException.File: EQCASLicensingSubSystem.cpp'
[FATAL][2017/05/03 21:31:00.535][0x00003ca8][0000059c][][ActivationSubSystem]
[GetActivationForSystemID][0] 'ActivationException.Line: 3999'
```

您可以为 Pattern 键/值对指定以下正则表达式，以帮助将日志文件分解为单条日志记录：

```
^\[\w+\]\[(?<TimeStamp>\d{4}/\d{2}/\d{2} \d{2}:\d{2}:\d{2}\.\d{3})\]
```

此正则表达式匹配以下序列：

1. 要评估的字符串的开头。
2. 由方括号括起的一个或多个单词字符。
3. 由方括号括起的时间戳。时间戳匹配以下序列：
   a. 四位数年份
   b. 正斜杠
   c. 两位数月份
   d. 正斜杠
   e. 两位数日期

f. 空格字符
g. 两位数小时
h. 冒号
i. 两位数分钟
j. 冒号
k. 两位数秒
l. 句点
m. 三位数毫秒

您可以为 TimestampFormat 键/值对指定以下格式，以将文本时间戳转换为日期和时间：

```
yyyy/MM/dd HH:mm:ss.fff
```

您可以使用以下正则表达式通过 ExtractionPattern 键/值对提取日志记录的字段。

```
^\[(?<Severity>\w+)\]\[(?<TimeStamp>\d{4}/\d{2}/\d{2} \d{2}:\d{2}:\d{2}\.
\d{3})\]\[[^]]*\]\[[^]]*\]\[[^]]*\]\[(?<SubSystem>\w+)\]\[(?<Module>\w+)\]\[[^]]*\] '(?
<Message>.*)'$
```

此正则表达式匹配序列中的以下组：

1. Severity— 用方括号括起的一个或多个单词字符。
2. TimeStamp— 参阅时间戳的上一描述。
3. 跳过三个包含 0 个或多个字符的用方括号括起的未命名序列。
4. SubSystem— 用方括号括起的一个或多个单词字符。
5. Module— 用方括号括起的一个或多个单词字符。
6. 跳过一个包含 0 个或多个字符的用方括号括起的未命名序列。
7. 跳过一个未指定空格。
8. Message— 用单引号括起的 0 个或多个字符。

以下源声明将这些正则表达式和日期时间格式组合在一起，为 Windows 的 Kinesis 代理提供了有关解析此类日志文件的完整说明。

```
{
```

```
    "Id": "PrintLog",
    "SourceType": "DirectorySource",
    "Directory": "C:\\temp\\PrintLogTest",
    "FileNameFilter": "*.log",
    "RecordParser": "Regex",
    "TimestampFormat": "yyyy/MM/dd HH:mm:ss.fff",
    "Pattern": "^\\[\\w+\\]\\[(?<TimeStamp>\\d{4}/\\d{2}/\\d{2} \\d{2}:\\d{2}:\\d{2}\\.
\\d{3})\\]",
    "ExtractionPattern": "^\\[(?<Severity>\\w+)\\]\\[(?<TimeStamp>\\d{4}/\\d{2}/\\d{2}
 \\d{2}:\\d{2}:\\d{2}\\.\\d{3})\\]\\[[^]]*\\]\\[[^]]*\\]\\[[^]]*\\]\\[(?<SubSystem>\\w
+)\\]\\[(?<Module>\\w+)\\]\\[[^]]*\\] '(?<Message>.*)'$",
    "TimeZoneKind": "UTC"
}
```

> **Note**
>
> 必须使用额外的反斜杠对 JSON 格式的文件中的反斜杠进行转义。

有关正则表达式的更多信息，请参阅 Microsoft .NET 文档中的正则表达式语言 – 快速参考。

`Delimited` 记录解析程序

您可以使用 `Delimited` 记录解析程序来解析半结构化日志和数据文件，其中存在一致的字符序列，用于分隔每行数据中的每列数据。例如，CSV 文件使用逗号分隔每列数据，TSV 文件使用选项卡。

假设您要解析由网络策略服务器生成的 Microsoft NPS 数据库格式的日志文件。此类文件可能如下所示：

```
"NPS-
MASTER","IAS",03/22/2018,23:07:55,1,"user1","Domain1\user1",,,,,,,,0,"192.168.86.137","Nate
 - Test 1",,,,,,,1,,0,"311 1 192.168.0.213 03/15/2018 08:14:29
 1",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,"Use Windows authentication for all users",1,,,,
"NPS-
MASTER","IAS",03/22/2018,23:07:55,3,,"Domain1\user1",,,,,,,,0,"192.168.86.137","Nate
 - Test 1",,,,,,,1,,16,"311 1 192.168.0.213 03/15/2018 08:14:29
 1",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,"Use Windows authentication for all users",1,,,,
```

以下示例 `appsettings.json` 配置文件包含一个 `DirectorySource` 声明，该声明使用 `Delimited` 记录解析程序来将此文本解析为对象表示形式。然后，它将 JSON 格式的数据流式传输到 Kinesis Data Firehose：

```
{
    "Sources": [
        {
            "Id": "NPS",
            "SourceType": "DirectorySource",
            "Directory": "C:\\temp\\NPS",
            "FileNameFilter": "*.log",
            "RecordParser": "Delimited",
            "Delimiter": ",",
            "Headers": "ComputerName,ServiceName,Record-Date,Record-Time,Packet-
Type,User-Name,Fully-Qualified-Distinguished-Name,Called-Station-ID,Calling-Station-
ID,Callback-Number,Framed-IP-Address,NAS-Identifier,NAS-IP-Address,NAS-Port,Client-
Vendor,Client-IP-Address,Client-Friendly-Name,Event-Timestamp,Port-Limit,NAS-Port-
Type,Connect-Info,Framed-Protocol,Service-Type,Authentication-Type,Policy-Name,Reason-
Code,Class,Session-Timeout,Idle-Timeout,Termination-Action,EAP-Friendly-Name,Acct-
Status-Type,Acct-Delay-Time,Acct-Input-Octets,Acct-Output-Octets,Acct-Session-Id,Acct-
Authentic,Acct-Session-Time,Acct-Input-Packets,Acct-Output-Packets,Acct-Terminate-
Cause,Acct-Multi-Ssn-ID,Acct-Link-Count,Acct-Interim-Interval,Tunnel-Type,Tunnel-
Medium-Type,Tunnel-Client-Endpt,Tunnel-Server-Endpt,Acct-Tunnel-Conn,Tunnel-Pvt-
Group-ID,Tunnel-Assignment-ID,Tunnel-Preference,MS-Acct-Auth-Type,MS-Acct-EAP-Type,MS-
RAS-Version,MS-RAS-Vendor,MS-CHAP-Error,MS-CHAP-Domain,MS-MPPE-Encryption-Types,MS-
MPPE-Encryption-Policy,Proxy-Policy-Name,Provider-Type,Provider-Name,Remote-Server-
Address,MS-RAS-Client-Name,MS-RAS-Client-Version",
            "TimestampField": "{Record-Date} {Record-Time}",
            "TimestampFormat": "MM/dd/yyyy HH:mm:ss"
        }
    ],
    "Sinks": [
        {
            "Id": "npslogtest",
            "SinkType": "KinesisFirehose",
            "Region": "us-west-2",
            "StreamName": "npslogtest",
            "Format": "json"
        }
    ],
    "Pipes": [
        {
            "Id": "W3SVCLog1ToKinesisStream",
            "SourceRef": "NPS",
            "SinkRef": "npslogtest"
        }
    ]
```

```
}
```

流式传输到 Kinesis 数据防火器的 JSON 格式的数据如下所示：

```
{
    "ComputerName": "NPS-MASTER",
    "ServiceName": "IAS",
    "Record-Date": "03/22/2018",
    "Record-Time": "23:07:55",
    "Packet-Type": "1",
    "User-Name": "user1",
    "Fully-Qualified-Distinguished-Name": "Domain1\\user1",
    "Called-Station-ID": "",
    "Calling-Station-ID": "",
    "Callback-Number": "",
    "Framed-IP-Address": "",
    "NAS-Identifier": "",
    "NAS-IP-Address": "",
    "NAS-Port": "",
    "Client-Vendor": "0",
    "Client-IP-Address": "192.168.86.137",
    "Client-Friendly-Name": "Nate - Test 1",
    "Event-Timestamp": "",
    "Port-Limit": "",
    "NAS-Port-Type": "",
    "Connect-Info": "",
    "Framed-Protocol": "",
    "Service-Type": "",
    "Authentication-Type": "1",
    "Policy-Name": "",
    "Reason-Code": "0",
    "Class": "311 1 192.168.0.213 03/15/2018 08:14:29 1",
    "Session-Timeout": "",
    "Idle-Timeout": "",
    "Termination-Action": "",
    "EAP-Friendly-Name": "",
    "Acct-Status-Type": "",
    "Acct-Delay-Time": "",
    "Acct-Input-Octets": "",
    "Acct-Output-Octets": "",
    "Acct-Session-Id": "",
    "Acct-Authentic": "",
    "Acct-Session-Time": "",
```

```
    "Acct-Input-Packets": "",
    "Acct-Output-Packets": "",
    "Acct-Terminate-Cause": "",
    "Acct-Multi-Ssn-ID": "",
    "Acct-Link-Count": "",
    "Acct-Interim-Interval": "",
    "Tunnel-Type": "",
    "Tunnel-Medium-Type": "",
    "Tunnel-Client-Endpt": "",
    "Tunnel-Server-Endpt": "",
    "Acct-Tunnel-Conn": "",
    "Tunnel-Pvt-Group-ID": "",
    "Tunnel-Assignment-ID": "",
    "Tunnel-Preference": "",
    "MS-Acct-Auth-Type": "",
    "MS-Acct-EAP-Type": "",
    "MS-RAS-Version": "",
    "MS-RAS-Vendor": "",
    "MS-CHAP-Error": "",
    "MS-CHAP-Domain": "",
    "MS-MPPE-Encryption-Types": "",
    "MS-MPPE-Encryption-Policy": "",
    "Proxy-Policy-Name": "Use Windows authentication for all users",
    "Provider-Type": "1",
    "Provider-Name": "",
    "Remote-Server-Address": "",
    "MS-RAS-Client-Name": "",
    "MS-RAS-Client-Version": ""
}
```

SysLog 记录解析程序

对于 SysLog 记录解析程序，源的解析输出包含以下信息：

属性	类型	描述
SysLogTimeStamp	字符串	syslog 格式的日志文件中的原始日期和时间。
Hostname	字符串	syslog 格式的日志文件所在的计算机的名称。

属性	类型	描述
Program	字符串	生成日志文件的应用程序或服务的名称。
Message	字符串	应用程序或服务生成的日志消息。
TimeStamp	字符串	ISO 8601 格式的解析的日期和时间。

以下是转换为 JSON 的 SysLog 数据的示例：

```
{
    "SysLogTimeStamp": "Jun 18 01:34:56",
    "Hostname": "myhost1.example.mydomain.com",
    "Program": "mymailservice:",
    "Message": "Info: ICID 123456789 close",
    "TimeStamp": "2017-06-18T01:34.56.000"
}
```

Summary

以下是可用于 DirectorySource 源的键/值对以及与这些键/值对相关的 RecordParser 的摘要。

密钥名称	RecordParser	备注
SourceType	对于所有项是必需的	必须具有值 DirectorySource
Directory	对于所有项是必需的	
FileNameFilter	对于所有项是可选的	
RecordParser	对于所有项是必需的	
TimestampField	对于 SingleLineJson 是可选的	

密钥名称	RecordParser	备注
TimestampFormat	对于 Timestamp 是必需的，对于 SingleLineJson 是可选的（如果指定 TimestampField ）	
Pattern	对于 Regex 是必需的	
ExtractionPattern	对于 Regex 是可选的	对于 Regex 是必需的（如果接收器指定 json 或 xml 格式）
Delimiter	对于 Delimited 是必需的	
HeaderPattern	对于 Delimited 是可选的	
Headers	对于 Delimited 是可选的	
RecordPattern	对于 Delimited 是可选的	
CommentPattern	对于 Delimited 是可选的	
TimeZoneKind	对于 Regex、Timestamp 、SysLog 和 SingleLineJson 是可选的（如果标识时间戳字段）	
SkipLines	对于所有项是可选的	

ExchangeLogSource 配置

ExchangeLogSource 类型用于从 Microsoft Exchange 收集日志。Exchange 以多种不同的日志格式生成日志。此源类型解析所有这些日志。虽然可以将 DirectorySource 类型与 Regex 记录解析程序结合使用来解析日志，但使用 ExchangeLogSource 要简单得多。这是因为您不需要为日志文件格式设计和提供正则表达式。以下是示例 ExchangeLogSource 声明：

```
{
   "Id": "MyExchangeLog",
   "SourceType": "ExchangeLogSource",
   "Directory": "C:\\temp\\ExchangeLogTest",
   "FileNameFilter": "*.log"
}
```

所有交换声明都可以提供以下键/值对：

SourceType

必须是文本字符串 "ExchangeLogSource"（必需）。

Directory

包含日志文件的目录的路径（必需）。

FileNameFilter

（可选）根据通配符文件命名模式限制从中收集日志数据的目录中的文件集。如果未指定此键/值对，则默认情况下会收集目录中所有文件的日志数据。

TimestampField

包含记录的日期和时间的列的名称。此键/值对是可选的，如果字段名为 date-time 或 DateTime，则无需指定它。否则，它是必需的。

W3SVCLogSource 配置

W3SVCLogSource 类型用于从 Internet Information Services (IIS) for Windows 收集日志。

以下是示例 W3SVCLogSource 声明：

```
{
   "Id": "MyW3SVCLog",
   "SourceType": "W3SVCLogSource",
```

```
    "Directory": "C:\\inetpub\\logs\\LogFiles\\W3SVC1",
    "FileNameFilter": "*.log"
}
```

所有 W3SVCLogSource 声明都可以提供以下键/值对：

SourceType

必须是文本字符串 "W3SVCLogSource"（必需）。

Directory

包含日志文件的目录的路径（必需）。

FileNameFilter

（可选）根据通配符文件命名模式限制从中收集日志数据的目录中的文件集。如果未指定此键/值对，则默认情况下会收集目录中所有文件的日志数据。

UlsSource 配置

UlsSource 类型用于从 Microsoft SharePoint 收集日志。以下是示例 UlsSource 声明：

```
{
    "Id": "UlsSource",
    "SourceType": "UlsSource",
    "Directory": "C:\\temp\\uls",
    "FileNameFilter": "*.log"
}
```

所有 UlsSource 声明都可以提供以下键/值对：

SourceType

必须是文本字符串 "UlsSource"（必需）。

Directory

包含日志文件的目录的路径（必需）。

FileNameFilter

（可选）根据通配符文件命名模式限制从中收集日志数据的目录中的文件集。如果未指定此键/值对，则默认情况下会收集目录中所有文件的日志数据。

WindowsEventLogSource 配置

WindowsEventLogSource 类型用于从 Windows Event Log 服务收集事件。以下是示例 WindowsEventLogSource 声明：

```
{
    "Id": "mySecurityLog",
    "SourceType": "WindowsEventLogSource",
    "LogName": "Security"
}
```

所有 WindowsEventLogSource 声明都可以提供以下键/值对：

SourceType

必须是文本字符串 "WindowsEventLogSource"（必需）。

LogName

从指定日志收集事件。常见值包括 Application、Security 和 System，但您可以指定任何有效的 Windows 事件日志名称。此键/值对是必需的。

Query

（可选）限制从 WindowsEventLogSource 输出的事件。如果未指定此键/值对，则默认情况下会输出所有事件。有关此值的语法的信息，请参阅 Windows 文档中的事件查询和事件 XML。有关日志级别定义的信息，请参阅 Windows 文档中的事件类型。

IncludeEventData

当此键/值对的值为 "true" 时，（可选）启用与指定 Windows 事件日志中的事件关联的特定于提供程序的事件数据的收集和流式传输。仅包括可成功序列化的事件数据。此键/值对是可选的，如果未指定它，则不会收集特定于提供程序的事件数据。

> **Note**
>
> 包含事件数据可能会显著增加从此源流式传输的数据量。事件的最大大小可以是 262143 字节（包括事件数据）。

从 WindowsEventLogSource 解析的输出包含以下信息：

属性	类型	描述
EventId	Int	事件类型的标识符。
Description	字符串	描述事件详细信息的文本。
LevelDisplayName	字符串	事件类别（“错误”、“警告”、“信息”、“成功审核”、“失败审核”之一）。
LogName	字符串	记录事件的位置（典型值为 Application 、Security 和 System，但有很多可能性）。
MachineName	字符串	记录了事件的计算机。
ProviderName	字符串	记录了事件的应用程序或服务。
TimeCreated	字符串	事件发生（用 ISO 8601 格式）的时间。
Index	Int	项在日志中的位置。
UserName	字符串	项的创建者（如果知道）。
Keywords	字符串	事件类型。标准值包括 AuditFailure （失败的安全审核事件）、AuditSuccess （成功的安全审核事件）、Classic（使用 RaiseEvent 函数引发的事件）、Correlation Hint（传输事件）、SQM（Service Quality Mechanism 事件）、WDI Context（Windows Diagnostic Infrastructure 上下文事件）和 WDI Diag（Windows Diagnostic Infrastructure 诊断事件）。
EventData	对象列表	有关日志事件的可选的特定于提供程序的额外数据。仅在 IncludeEv

属性	类型	描述
		entData 键/值对的值为 "true" 时包含此项。

以下是转换为 JSON 的示例事件：

```
{[
    "EventId": 7036,
    "Description": "The Amazon SSM Agent service entered the stopped state.",
    "LevelDisplayName": "Informational",
    "LogName": "System",
    "MachineName": "mymachine.mycompany.com",
    "ProviderName": "Service Control Manager",
    "TimeCreated": "2017-10-04T16:42:53.8921205Z",
    "Index": 462335,
    "UserName": null,
    "Keywords": "Classic",
    "EventData": [
    "Amazon SSM Agent",
    "stopped",
    "rPctBAMZFhYubF8zVLcrBd3bTTcNzHvY5Jc2Br0aMrxxx=="
]}
```

窗口七个日志轮询源配置

WindowsEventLogPollingSource使用基于轮询的机制从事件日志中收集与配置的参数匹配的所有新事件。根据上次轮询期间收集的事件数量，轮询间隔会在 100 ms 到 5000 ms 之间动态更新。以下是示例 WindowsEventLogPollingSource 声明：

```
{
    "Id": "MySecurityLog",
    "SourceType": "WindowsEventLogPollingSource",
    "LogName": "Security",
    "IncludeEventData": "true",
    "Query": "",
    "CustomFilters": "ExcludeOwnSecurityEvents"
}
```

所有 WindowsEventLogPollingSource 声明都可以提供以下键/值对：

`SourceType`

必须是文本字符串 `"WindowsEventLogPollingSource"`（必需）。

`LogName`

指定日志。有效选项为`Application`、`Security`、`System`或其他有效日志。

`IncludeEventData`

可选。何时`true`，指定以 JSON 和 XML 的形式流式传输时包含额外的 EventData。默认为`false`。

`Query`

可选。Windows 事件日志支持使用 XPath 表达式查询事件，您可以使用`Query`。有关更多信息，请参阅 。事件查询和事件 XML在微软文档中。

`CustomFilters`

可选。用分号隔开的筛选器列表 (`;`)。可以指定以下筛选器。

`ExcludeOwnSecurityEvents`

排除 Windows 本身的 Kinesis 代理生成的安全事件。

WindowsETWEventSource 配置

`WindowsETWEventSource` 类型用于通过名为 Windows 事件跟踪 (ETW) 的功能来收集应用程序和服务事件跟踪。有关更多信息，请参阅 Windows 文档中的事件跟踪。

以下是示例 `WindowsETWEventSource` 声明：

```
{
    "Id": "ClrETWEventSource",
    "SourceType": "WindowsETWEventSource",
    "ProviderName": "Microsoft-Windows-DotNETRuntime",
    "TraceLevel": "Verbose",
    "MatchAnyKeyword": 32768
}
```

所有 `WindowsETWEventSource` 声明都可以提供以下键/值对：

`SourceType`

必须是文本字符串 `"WindowsETWEventSource"`（必需）。

`ProviderName`

指定用于收集跟踪事件的事件提供程序。这必须是已安装的提供程序的有效 ETW 提供程序名称。要确定安装了哪些提供程序，请在 Windows 命令提示符窗口中执行以下命令：

```
logman query providers
```

`TraceLevel`

指定应收集哪些类别的跟踪事件。允许的值包括 `Critical`、`Error`、`Warning`、`Informational` 和 `Verbose`。确切含义取决于所选的 ETW 提供程序。

`MatchAnyKeyword`

该值是一个 64 位数，其中每个位代表一个单独的关键字。每个关键字都描述了要收集的事件类别。有关受支持的关键字及其值以及它们与 `TraceLevel` 的关联方式，请参阅该提供程序的文档。例如，有关 CLR ETW 提供程序的信息，请参阅 Microsoft .NET Framework 文档中的 CLR ETW 关键字和级别。

在前面的示例中，32768 (0x00008000) 表示 CLR ETW 提供程序的 `ExceptionKeyword`，它指示提供程序收集有关引发的异常的信息。虽然 JSON 本身不支持十六进制常量，但您可以通过将它们放在一个字符串中为 `MatchAnyKeyword` 指定它们。您也可以指定多个常量（用逗号分隔）。例如，使用以下命令指定 `ExceptionKeyword` 和 `SecurityKeyword` (0x00000400)：

```
{
   "Id": "MyClrETWEventSource",
   "SourceType": "WindowsETWEventSource",
   "ProviderName": "Microsoft-Windows-DotNETRuntime",
   "TraceLevel": "Verbose",
   "MatchAnyKeyword": "0x00008000, 0x00000400"
}
```

为了确保为提供程序启用所有指定的关键字，使用 OR 组合多个关键字值并将其传递给该提供程序。

来自 `WindowsETWEventSource` 的输出包含每个事件的以下信息：

属性	类型	描述
EventName	字符串	发生的事件的类型。
ProviderName	字符串	检测到事件的提供程序。
FormattedMessage	字符串	事件的文本摘要。
ProcessID	Int	报告了事件的过程。
ExecutingThreadID	Int	报告了事件的过程中的线程。
MachineName	字符串	报告事件的桌面设备或服务器的名称。
Payload	哈希表	具有字符串键和任何类型的对象作为值的表。键是负载项名称，值是负载项的值。负载依赖于提供程序。

以下是转换为 JSON 的示例事件：

```
{
     "EventName": "Exception/Start",
     "ProviderName": "Microsoft-Windows-DotNETRuntime",
     "FormattedMessage": "ExceptionType=System.Exception;\r
\nExceptionMessage=Intentionally unhandled exception.;\r\nExceptionEIP=0x2ab0499;\r
\nExceptionHRESULT=-2,146,233,088;\r\nExceptionFlags=CLSCompliant;\r\nClrInstanceID=9
 ",
     "ProcessID": 3328,
     "ExecutingThreadID": 6172,
     "MachineName": "MyHost.MyCompany.com",
     "Payload":
      {
        "ExceptionType": "System.Exception",
        "ExceptionMessage": "Intentionally unhandled exception.",
        "ExceptionEIP": 44762265,
        "ExceptionHRESULT": -2146233088,
        "ExceptionFlags": 16,
        "ClrInstanceID": 9
      }
```

```
}
```

WindowsPerformanceCounterSource 配置

WindowsPerformanceCounterSource 类型从 Windows 中收集性能计数器指标。以下是示例 WindowsPerformanceCounterSource 声明：

```
{
 "Id": "MyPerformanceCounter",
 "SourceType": "WindowsPerformanceCounterSource",
 "Categories": [{
   "Category": "Server",
   "Counters": ["Files Open", "Logon Total", "Logon/sec", "Pool Nonpaged Bytes"]
  },
  {
   "Category": "System",
   "Counters": ["Processes", "Processor Queue Length", "System Up Time"]
  },
  {
   "Category": "LogicalDisk",
   "Instances": "*",
   "Counters": [
    "% Free Space", "Avg. Disk Queue Length",
    {
     "Counter": "Disk Reads/sec",
     "Unit": "Count/Second"
    },
    "Disk Writes/sec"
   ]
  },
  {
   "Category": "Network Adapter",
   "Instances": "^Local Area Connection\* \d$",
   "Counters": ["Bytes Received/sec", "Bytes Sent/sec"]
  }
 ]
}
```

所有 WindowsPerformanceCounterSource 声明都可以提供以下键/值对：

SourceType

必须是文本字符串 "WindowsPerformanceCounterSource"（必需）。

`Categories`

指定要从 Windows 中收集的一组性能计数器指标组。每个指标组都包含以下键/值对：

`Category`

指定要收集的指标的计数器集（必需）。

`Instances`

当每个对象都有一组唯一的性能计数器时，指定感兴趣的对象集。例如，当类别为 `LogicalDisk` 时，每个磁盘驱动器都有一组性能计数器。键/值对是可选的。您可以使用通配符 * 和 ? 匹配多个实例。要聚合所有实例的值，请指定 `_Total`。

您还可以使用`InstanceRegex`，它接受包含*通配符作为实例名称的一部分。

`Counters`

指定要为指定类别收集的指标。此键/值对是必需的。您可以使用通配符 * 和 ? 匹配多个计数器。您可以仅使用名称或使用名称和单位来指定 `Counters`。如果未指定计数器单位，则 Windows 的 Kinesis 代理会尝试从名称推断单位。如果这些推理不正确，则可以明确指定单位。如果需要，您可以更改 `Counter` 名称。计数器的更复杂的表示形式是具有以下键/值对的对象：

`Counter`

计数器的名称。此键/值对是必需的。

`Rename`

要提供给接收器的计数器的名称。键/值对是可选的。

`Unit`

与计数器关联的值的含义。有关有效设备名称的完整列表，请参阅MetricDatum中的Amazon CloudWatch API 参考。

以下是复杂计数器规范的示例：

```
{
   "Counter": "Disk Reads/sec,
   "Rename": "Disk Reads per second",
   "Unit": "Count/Second"
```

```
}
```

`WindowsPerformanceCounterSource`只能与指定 Amazon CloudWatch 接收器的管道结合使用。如果 Windows Kinesis 代理内置指标也流式传输到 CloudWatch，请使用单独的接收器。检查适用于 Windows 的 Kinesis 代理日志，以确定在未指定单位时，为计数器推断了哪些单位。`WindowsPerformanceCounterSource`声明。使用 PowerShell 确定类别、实例和计数器的有效名称。

要查看有关所有类别的信息（包括与计数器集关联的计数器），请在 PowerShell 窗口中执行以下命令：

```
Get-Counter -ListSet * | Sort-Object
```

要确定计数器集中每个计数器可用的实例，请在 PowerShell 窗口中执行类似于以下示例的命令：

```
Get-Counter -Counter "\Process(*)\% Processor Time"
```

`Counter` 参数的值应是上一个 `Get-Counter -ListSet` 命令调用列出的 `PathsWithInstances` 成员中的路径之一。

Windows 内置指标源的 Kinesis 代理

除了普通指标来源（如`WindowsPerformanceCounterSource`类型（请参阅WindowsPerformanceCounterSource 配置），CloudWatch 接收器类型可以从收集有关适用于 Windows 本身的 Kinesis 代理的指标的特殊源接收指标。Windows 指标的 Kinesis 代理还可用于`KinesisTap`类别的 Windows 性能计数器。

这些区域有：`MetricsFilter`键 Kinesis 值对指定哪些指标从内置的 Windows 功能代理程序指标源流式传输到 CloudWatch。该值是一个字符串，其中包含一个或多个用分号分隔的筛选表达式；例如：

`"MetricsFilter": "FilterExpression1;FilterExpression2"`

与一个或多个筛选表达式匹配的指标将流式传输到 CloudWatch。

单实例指标本质上是全局的，不与特定的源或接收器关联。多实例指标是多维的，并且基于源或接收器声明 Id。每个源或接收器类型均可具有一组不同的指标。

有关 Windows 指标名称的内置 Kinesis 代理的列表，请参阅适用于 Windows 指标的 Kinesis 代理列表。

对于单实例指标，筛选表达式是指标的名称；例如：

```
"MetricsFilter": "SourcesFailedToStart;SinksFailedToStart"
```

对于多实例指标，筛选表达式依次包含指标名称、句点 (.) 和生成该指标的源或接收器声明的 Id。例如，假设有一个 Id 为 MyFirehose 的接收器声明：

```
"MetricsFilter": "KinesisFirehoseRecordsFailedNonrecoverable.MyFirehose"
```

您可以使用旨在区分单实例指标和多实例指标的特殊通配符模式。

- 星号 (*) 将匹配 0 个或多个字符，句点 (.) 除外。
- 问号 (?) 匹配一个字符，句点除外。
- 任何其他字符仅匹配自身。
- _Total 是一个特殊标记，它会导致跨维度聚合所有匹配的多实例值。

以下示例匹配所有单实例指标：

```
"MetricsFilter": "*"
```

由于星号与句点字符不匹配，因此，仅包含单实例指标。

以下示例匹配所有多实例指标：

```
"MetricsFilter": "*.*"
```

以下示例匹配所有指标（单个和多个）：

```
"MetricsFilter": "*;*.*"
```

以下示例跨所有源和接收器聚合所有多实例指标：

```
"MetricsFilter": "*._Total"
```

以下示例聚合所有 Kinesis Data Firehose 收器的所有 Kinesis 数据防火管指标：

```
"MetricsFilter": "*Firehose*._Total"
```

以下示例匹配所有单实例和多实例错误指标：

```
"MetricsFilter": "*Failed*;*Error*.*;*Failed*.*"
```

以下示例匹配跨所有源和接收器聚合的所有不可恢复的错误指标：

```
"MetricsFilter": "*Nonrecoverable*._Total"
```

有关如何指定使用适用于 Windows 的 Kinesis 代理的内置指标源的信息，请参阅为 Windows 度量管道配置 Kinesis 代理。

适用于 Windows 指标的 Kinesis 代理列表

以下是对适用于 Windows 的 Kinesis 代理可用的单实例指标和多实例指标的列表。

单实例指标

提供了以下单实例指标：

`KinesisTapBuildNumber`

Windows 的 Kinesis 代理的版本号。

`PipesConnected`

已成功将其源连接到其接收器的管道的数目。

`PipesFailedToConnect`

已成功将其源连接到其接收器的管道的数目。

`SinkFactoriesFailedToLoad`

未成功加载到 Windows 的 Kinesis 代理的接收器类型的数目。

`SinkFactoriesLoaded`

已成功加载到 Windows Kinesis 代理的接收器类型的数目。

`SinksFailedToStart`

未成功启动（通常是因接收器声明不正确导致的）的接收器的数目。

`SinksStarted`

已成功启动的接收器的数目。

`SourcesFailedToStart`

未成功启动（通常是因源声明不正确导致的）的源的数目。

`SourcesStarted`

已成功启动的源的数目。

`SourceFactoriesFailedToLoad`

未成功加载到 Windows 的 Kinesis 代理的源类型的数目。

`SourceFactoriesLoaded`

已成功加载到 Windows Kinesis 代理的源类型的数目。

多实例指标

提供了以下多实例指标：

DirectorySource 指标

`DirectorySourceBytesRead`

在此 `DirectorySource` 的间隔期间读取的字节的数目。

`DirectorySourceBytesToRead`

已知可供读取但尚未由 Kinesis 代理读取的字节数（适用于 Windows）。

`DirectorySourceFilesToProcess`

需要检查但尚未由 Kinesis 代理的 Windows 检查的已知文件的数目。

DirectorySourceRecordsRead

在此 DirectorySource 的间隔期间已读取的记录数。

WindowsEventLogSource 指标

EventLogSourceEventsError

未成功读取的 Windows 事件日志事件的数目。

EventLogSourceEventsRead

已成功读取的 Windows 事件日志事件的数目。

KinesisFirehose 接收器指标

KinesisFirehoseBytesAccepted

间隔期间已接受的字节数。

KinesisFirehoseClientLatency

记录生成和记录流式传输到 Kinesis Data Firehose 服务之间经过的时间。

KinesisFirehoseLatency

Kinesis Data Firehose 服务的记录流的开始和结束之间经过的时间。

KinesisFirehoseNonrecoverableServiceErrors

无法无错误地将记录发送到 Kinesis 数据消防软管服务的次数（尽管已重试）。

KinesisFirehoseRecordsAttempted

已尝试流式传输到 Kinesis 数据消防软管服务的记录的数目。

KinesisFirehoseRecordsFailedNonrecoverable

未成功流式传输到 Kinesis 数据消防软管服务的记录的数目（尽管已重试）。

KinesisFirehoseRecordsFailedRecoverable

已成功流式传输到 Kinesis 数据消防软管服务的记录的数目（但仅在重试的情况下）。

KinesisFirehoseRecordsSuccess

已成功流式传输到 Kinesis 数据防火管服务的记录的数目（无需重试）。

`KinesisFirehoseRecoverableServiceErrors`

将记录成功发送到 Kinesis 数据消防软管服务的次数（但仅在重试的情况下）。

KinesisStream 指标

`KinesisStreamBytesAccepted`

间隔期间已接受的字节数。

`KinesisStreamClientLatency`

记录生成和记录流式传输到 Kinesis Data Streams 服务之间经过的时间。

`KinesisStreamLatency`

Kinesis 数据流服务的记录流的开始和结束之间经过的时间。

`KinesisStreamNonrecoverableServiceErrors`

无法无错误地将记录发送到 Kinesis 数据流服务的次数（尽管已重试）。

`KinesisStreamRecordsAttempted`

已尝试流式传输到 Kinesis 数据流服务的记录的数目。

`KinesisStreamRecordsFailedNonrecoverable`

未成功流式传输到 Kinesis 数据流服务的记录的数目（尽管已重试）。

`KinesisStreamRecordsFailedRecoverable`

已成功流式传输到 Kinesis 数据流服务的记录的数目（但仅在重试的情况下）。

`KinesisStreamRecordsSuccess`

已成功流式传输到 Kinesis Data Streams 服务的记录的数目。

`KinesisStreamRecoverableServiceErrors`

将记录成功发送到 Kinesis 数据流服务的次数（但仅在重试的情况下）。

CloudWatchLog 指标

`CloudWatchLogBytesAccepted`

间隔期间已接受的字节数。

`CloudWatchLogClientLatency`

记录生成和记录流式传输到 CloudWatch Logs 服务之间经过的时间。

`CloudWatchLogLatency`

CloudWatch Logs 服务的记录流的开始和结束之间经过的时间。

`CloudWatchLogNonrecoverableServiceErrors`

无法无错误地将记录发送到 CloudWatch Logs 服务的次数（尽管已重试）。

`CloudWatchLogRecordsAttempted`

已尝试流式传输到 CloudWatch Logs 服务的记录的数目。

`CloudWatchLogRecordsFailedNonrecoverable`

未成功流式传输到 CloudWatch Logs 服务的记录的数目（尽管已重试）。

`CloudWatchLogRecordsFailedRecoverable`

已成功流式传输到 CloudWatch Logs 服务的记录的数目（但仅在重试的情况下）。

`CloudWatchLogRecordsSuccess`

已成功流式传输到 CloudWatch Logs 服务的记录的数目。

`CloudWatchLogRecoverableServiceErrors`

将记录成功发送到 CloudWatch Logs 服务的次数（但仅在重试的情况下）。

CloudWatch 指标

`CloudWatchLatency`

CloudWatch 服务的指标流的开始和结束之间平均经过的时间。

`CloudWatchNonrecoverableServiceErrors`

无法无错误地将指标发送到 CloudWatch 服务的次数（尽管已重试）。

`CloudWatchRecoverableServiceErrors`

无错误地将指标发送到 CloudWatch 服务的次数（但仅在重试的情况下）。

`CloudWatchServiceSuccess`

无错误地将指标发送到 CloudWatch 服务的次数（无需重试）。

书签配置

默认情况下，Windows 的 Kinesis 代理将日志记录发送到在代理启动后创建的接收器。有时，发送早期日志记录很有用，例如，当在自动更新期间停止时创建的日志记录。书签功能可跟踪已发送到接收器的记录。当 Windows 的 Kinesis 代理处于书签模式中并启动时，它会将在 Windows 的 Kinesis 代理停止后创建的所有日志记录与随后创建的任何日志记录一起发送。要控制此行为，基于文件的源声明可以选择包括以下键/值对：

`InitialPosition`

指定书签的初始情况。可能值如下所示：

`EOS`

指定流结束 (EOS)。仅将在代理运行时创建的日志记录发送到接收器。

`0`

最初发送所有可用的日志记录和事件。然后，创建书签以确保最终将发送在创建书签后创建的每条新日志记录和事件，无论是否正在运行。

`Bookmark`

书签初始化为最新的日志记录或事件之后的书签。然后，创建书签以确保最终将发送在创建书签后创建的每条新日志记录和事件，无论是否正在运行。

默认情况下已启用书签。文件存储在`%ProgramData%\Amazon\KinesisTap`目录。

`Timestamp`

发送在 `InitialPositionTimestamp` 值（定义如下）之后创建的日志记录和事件。然后，创建书签以确保最终将发送在创建书签后创建的每条新日志记录和事件，无论是否正在运行。

`InitialPositionTimestamp`

指定所需的最早日志记录或事件时间戳。仅在 `InitialPosition` 的值为 `Timestamp` 时指定此键/值对。

`BookmarkOnBufferFlush`

可以将此设置添加到任何可书签源。如果设置为`true`，确保书签更新仅在接收器成功向 AWS 发送事件时才进行。您只能为源订阅单个汇。如果您要将日志发送到多个目的地，请复制源以避免数据丢失的潜在问题。

当 Windows 的 Kinesis 代理已停止很长一段时间，可能需要删除这些书签，因为已添加书签的日志记录和事件可能不再存在。给定源 ID 的书签文件位于 %PROGRAMDATA%\Amazon\AWSKinesisTap*source id*.bm。

书签不适用于已重命名或截断的文件。由于 ETW 事件和性能计数器的性质，无法为它们添加书签。

接收器声明

接收器声明 指定应从哪里以何种格式将日志、事件和指标发送到各种 AWS 服务。以下部分描述了 Microsoft Windows 的 Amazon Kinesis 代理中可用的内置接收器类型的配置。由于 Windows 的 Kinesis 代理是可扩展的，因此，您可以添加自定义接收器类型。通常在与各个接收器类型关联的配置声明中，需要有该接收器类型的唯一键/值对。

所有接收器声明可以包含以下键/值对：

`Id`

唯一字符串，在配置文件中标识特定接收器（必需）。

`SinkType`

此接收器的接收器类型的名称（必需）。接收器类型指定此接收器流式传输的日志、事件或指标数据的目标。

`AccessKey`

指定在授权访问与接收器类型关联的 AWS 服务时，使用的 AWS 访问密钥。键/值对是可选的。有关更多信息，请参阅 接收器安全配置。

`SecretKey`

指定在授权访问与接收器类型关联的 AWS 服务时，使用的 AWS 私有密钥。键/值对是可选的。有关更多信息，请参阅 接收器安全配置。

`Region`

指定哪个 AWS 区域包含用于流式传输的目标资源。键/值对是可选的。

`ProfileName`

指定用于身份验证的 AWS 配置文件。键/值对是可选的，但如果指定，则会覆盖任何指定的访问密钥和私有密钥。有关更多信息，请参阅 接收器安全配置。

`RoleARN`

指定在访问与接收器类型关联的 AWS 服务时使用的 IAM 角色。当 Windows Kinesis Agent 在 EC2 实例上运行但其他角色比实例配置文件所引用的角色更合适时，此选项非常有用。例如，跨账户角色可用于定位与 EC2 实例不在相同 AWS 账户中的资源。键/值对是可选的。

`Format`

指定在流式传输之前，应用到日志和事件数据的序列化类型。有效值为 `json` 和 `xml`。当数据管道中的下游分析需要或偏好特定格式的数据时，此选项非常有用。此键/值对可选，如果未指定，则将来自源的普通文本从接收器流式传输到与该接收器类型关联的 AWS 服务。

`TextDecoration`

未指定 `Format` 时，`TextDecoration` 指定在流式传输日志或事件记录时应包含的其他文本。有关更多信息，请参阅 配置接收器修饰。键/值对是可选的。

`ObjectDecoration`

指定了 `Format` 时，`ObjectDecoration` 指定在序列化和流式传输之前，日志或事件记录中包含哪些其他数据。有关更多信息，请参阅 配置接收器修饰。键/值对是可选的。

`BufferInterval`

为了尽可能减少对与接收器类型关联的 AWS 服务的 API 调用，Windows Kinesis Agent 在流式传输之前缓冲多个日志、事件或指标记录。这可以为针对每次 API 调用收费的服务节省费用。`BufferInterval` 指定在流式传输到 AWS 服务之前，缓冲记录的最大时间长度（以秒为单位）。此键/值对可选，在指定时，使用字符串来表示值。

`BufferSize`

为了尽可能减少对与接收器类型关联的 AWS 服务的 API 调用，Windows Kinesis Agent 在流式传输之前缓冲多个日志、事件或指标记录。这可以为针对每次 API 调用收费的服务节省费用。`BufferSize` 指定在流式传输到 AWS 服务之前，缓冲的最大记录数。此键/值对可选，在指定时，使用字符串来表示值。

`MaxAttempts`

指定在流式传输持续失败时，Windows Kinesis Agent 尝试将一组日志、事件和指标记录流式传输到 AWS 服务的次数。键/值对是可选的。在指定时，使用字符串来表示值。默认值为“3”。

有关使用不同接收器类型的完整配置文件示例，请参阅从 Windows 应用程序事件日志流式传输到接收器。

主题

KinesisStream 接收器配置

这些区域有：KinesisStream接收器类型将日志记录和事件流式传输到 Kinesis Data Streams 服务。通常，流式传输到 Kinesis Data Tata Stream 的数据由一个或多个使用各种 AWS 服务执行的自定义应用程序处理。数据流式传输到使用 Kinesis 数据流配置的指定流。有关更多信息，请参阅 。Amazon Kinesis Data Streams 开发人员指南。

以下是 Kinesis Data Streams 接收器声明的示例：

```
{
    "Id": "TestKinesisStreamSink",
    "SinkType": "KinesisStream",
    "StreamName": "MyTestStream",
    "Region": "us-west-2"
}
```

所有 KinesisStream 接收器声明可以提供以下额外的键/值对：

SinkType

必须指定，值必须为文本字符串 KinesisStream。

StreamName

指定 Kinesis 数据流的名称，该数据流接收从KinesisStream汇类型（必需）。在流式传输数据之前，请在 AWS 管理控制台、AWS CLI 或通过使用 Kinesis Data Street API 的应用程序来配置流。

RecordsPerSecond

指定每秒流式传输到 Kinesis Data Streams 的最大记录数。键/值对是可选的。如果指定，则使用整数来表示值。默认值为 1000 个记录。

BytesPerSecond

指定每秒流式传输到 Kinesis Data Streams 的最大字节数。键/值对是可选的。如果指定，则使用整数来表示值。默认值为 1 MB。

此接收器类型的默认 BufferInterval 为 1 秒，默认 BufferSize 为 500 个记录。

KinesisFirehose 接收器配置

这些区域有：KinesisFirehose接收器类型将日志记录和事件流式传输到 Kinesis Data Firehose 服务。Kinesis Data Firehose 将流式传输的数据提供给其他服务进行存储。通常，接下来在数据管道的后续阶段中会分析存储的数据。数据流式传输到使用 Kinesis Data Firehose 配置的指定传输流。有关更多信息，请参阅 。Amazon Kinesis Data Firehose 开发人员指南。

以下是 Kinesis 数据消防器接收器声明的示例：

```
{
   "Id": "TestKinesisFirehoseSink",
   "SinkType": "KinesisFirehose",
   "StreamName": "MyTestFirehoseDeliveryStream",
   "Region": "us-east-1",
   "CombineRecords": "true"
}
```

所有 KinesisFirehose 接收器声明可以提供以下额外的键/值对：

SinkType

必须指定，值必须为文本字符串 KinesisFirehose。

StreamName

指定 Kinesis Data Firehose 传输流的数据流名称，该流接收从KinesisStream汇类型（必需）。在流式传输数据之前，请使用 AWS 管理控制台、AWS CLI 或通过使用 Kinesis Data Firehose API 的应用程序来配置传输流。

CombineRecords

如果设置为true，指定将多个小记录合并成一个最大大小为 5 KB 的大记录。键/值对是可选的。使用此函数组合的记录由\n。如果您使用 AWS Lambda 转换 Kinesis Data Firehose 记录，则 Lambda 函数需要考虑分隔符字符。

RecordsPerSecond

指定每秒流式传输到 Kinesis Data Streams 的最大记录数。键/值对是可选的。如果指定，则使用整数来表示值。默认值为 5000 条记录。

BytesPerSecond

指定每秒流式传输到 Kinesis Data Streams 的最大字节数。键/值对是可选的。如果指定，则使用整数来表示值。默认值为 5 MB。

此接收器类型的默认 BufferInterval 为 1 秒，默认 BufferSize 为 500 个记录。

CloudWatch 接收器配置

这些区域有：CloudWatch接收器类型将指标流式传输到 CloudWatch Service。您可在 AWS 管理控制台中查看指标。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

以下为 CloudWatch 接收器声明：

```
{
   "Id": "CloudWatchSink",
   "SinkType": "CloudWatch"
}
```

所有 CloudWatch 接收器声明可以提供以下额外的键/值对：

SinkType

必须指定，值必须为文本字符串 CloudWatch。

Interval

指定 Windows Kinesis 代理程序向 CloudWatch 服务报告指标的频率（以秒为单位）。键/值对是可选的。如果指定，则使用整数来表示值。默认值为 60 秒。如果您希望高分辨率的 CloudWatch 指标，则指定 1 秒。

Namespace

指定指标数据报告到的 CloudWatch 命名空间。CloudWatch 命名空间将指标集分组到一起。键/值对是可选的。默认值为 KinesisTap。

Dimensions

指定用于在命名空间中隔离指标集的 CloudWatch 维度。这会非常有用，例如，为各个台式机或服务器提供单独的指标集数据。此键/值对可选，在指定时，其值必须符合以下格式："key1=value1;key2=value2..."。默认值为 "ComputerName={computername};InstanceId={instance_id}"。此值支持接收器变量替换。有关更多信息，请参阅 配置接收器变量替换。

MetricsFilter

指定哪些指标将哪些指标从内置的 Windows Kinesis 代理程序指标源流式传输到 CloudWatch。有关 Windows 指标源的内置 Kinesis Agent 的详细信息，包括此键/值对的值的语法详细信息，请参阅Windows 内置指标源的 Kinesis 代理。

CloudWatchLogs 接收器配置

这些区域有：CloudWatchLogs接收器类型将日志记录和事件流式传输到 Amazon CloudWatch Logs。您可在 AWS 管理控制台中查看日志，或者通过数据管道的其他阶段处理它们。数据流式传输到在 CloudWatch 日志中配置的指定日志流。日志流按照指定的日志组排列。有关更多信息，请参阅 。Amazon CloudWatch Logs 用户指南。

以下是 CloudWatch Logs 接收器声明的示例：

```
{
```

```
    "Id": "MyCloudWatchLogsSink",
    "SinkType": "CloudWatchLogs",
    "BufferInterval": "60",
    "BufferSize": "100",
    "Region": "us-west-2",
    "LogGroup": "MyTestLogGroup",
    "LogStream": "MyTestStream"
}
```

所有 CloudWatchLogs 接收器声明必须提供以下额外的键/值对：

SinkType

必须为文本字符串 CloudWatchLogs。

LogGroup

指定 CloudWatch Logs Logs 日志组的名称，该日志组包含的日志流接收由CloudWatchLogs汇类型。如果指定的日志组不存在，Windows 的 Kinesis 代理将尝试创建它。

LogStream

指定 CloudWatch Logs 流的名称，该日志流由CloudWatchLogs汇类型。此值支持接收器变量替换。有关更多信息，请参阅 配置接收器变量替换。如果指定的日志流不存在，Windows 的 Kinesis 代理将尝试创建它。

此接收器类型的默认 BufferInterval 为 1 秒，默认 BufferSize 为 500 个记录。最大缓冲区大小为 10000 个记录。

本地FileSystem接收器配置

汇类型FileSystem将日志和事件记录保存到本地文件系统上的文件中，而不是将其流式传输到 AWS 服务。FileSystem汇对于测试和诊断非常有用。例如，您可以使用此接收器类型在将记录发送到 AWS 之前检查记录。

与FileSystem接收器，您还可以使用配置参数来模拟批处理、限制和错误时重试，以模拟实际 AWS 汇的行为。

来自所有源的所有记录连接到FileSystem接收器保存到指定为FilePath。如果FilePath未指定，则会将记录保存到名为*SinkId*.txt中的%TEMP%目录，通常是C:\Users*UserName*\AppData\Local\Temp，其中，*SinkId*是接收器的唯一标识符，*UserName*是活动用户的 Windows 用户名。

此接收器类型支持文本装饰属性。有关更多信息，请参阅 配置接收器修饰。

一个示例FileSystem接收器类型配置显示在以下示例中。

```
{
    "Id": "LocalFileSink",
    "SinkType": "FileSystem",
    "FilePath": "C:\\ProgramData\\Amazon\\local_sink.txt",
    "Format": "json",
    "TextDecoration": "",
    "ObjectDecoration": ""
}
```

这些区域有：FileSystem配置由以下键/值对组成。

SinkType

必须为文本字符串 FileSystem。

FilePath

指定保存记录的路径和文件。键/值对是可选的。如果未指定，默认值为*TempPath**SinkId*.txt，其中，*TempPath*是存储在%TEMP%变量和*SinkId*是接收器的唯一标识符。

Format

将事件的格式指定为json或者xml。此键值对是可选的，不区分大小写。如果省略，事件将以纯文本形式写入文件。

TextDecoration

仅适用于以纯文本形式编写的事件。键/值对是可选的。

ObjectDecoration

仅适用于Format设置为json。键/值对是可选的。

高级使用 — 记录限制和故障模拟

FileSystem可以通过模拟记录限制来模拟 AWS 汇的行为。可使用以下键-值对指定记录限制和失效模拟属性。

通过获取目标文件的锁定并阻止对其进行写入，您可以使用FileSystem汇来模拟和检查网络发生故障时 AWS 汇的行为。

以下示例显示了一个FileSystem配置与模拟属性。

```
{
    "Id": "LocalFileSink",
    "SinkType": "FileSystem",
    "FilePath": "C:\\ProgramData\\Amazon\\local_sink.txt",
    "TextDecoration": "",
    "RequestsPerSecond": "100",
    "BufferSize": "10",
    "MaxBatchSize": "1024"
}
```

RequestsPerSecond

可选并指定为字符串类型。如果省略，默认值为"5"。控制接收器处理的请求速率（即写入文件），而不是记录数。适用于 Windows 的 Kinesis 代理向 AWS 终端节点发出批量请求，因此一个请求可能包含多个记录。

BufferSize

可选并指定为字符串类型。指定在保存到文件之前，接收器批处理的最大事件记录数。

MaxBatchSize

可选并指定为字符串类型。指定接收器在保存到文件之前批处理的最大事件记录数据量（以字节为单位）。

最大记录速率限制是BufferSize，它确定每个请求的最大记录数，RequestsPerSecond。您可以使用以下公式计算每秒记录速率限制。

记录率=BufferSize*RequestsPerSecond

给定上面示例中的配置值，最大记录速率为每秒 1000 条记录。

接收器安全配置

配置身份验证

对于 Windows Kinesis Agent 将日志、事件和指标流式传输到 AWS 服务，必须对访问进行身份验证。有多种方法可以为 Windows 提供身份验证。如何提供身份验证取决于 Windows 的 Kinesis Agent 以及特定组织的特定安全要求。

- 如果 Windows Kinesis Agent for Windows 在 Amazon EC2 主机上执行，则提供身份验证最简单安全的方法是，创建对所需 AWS 服务的所需操作具有足够访问权限的 IAM 角色，以及引用该角色的 EC2 实例配置文件。有关创建实例配置文件的信息，请参阅使用实例配置文件。有关将哪些策略附加到 IAM 角色的信息，请参阅配置授权。

  创建实例配置文件之后，您可以将其与使用适用于 Windows 的 Kinesis Agent 的任意 EC2 实例关联。如果实例已有关联的实例配置文件，您可以将相应的策略附加到与该实例配置文件关联的角色。
- 如果 Windows Kinesis Agent for Windows 在一个账户中的 EC2 主机上执行，但作为接收器目标的资源位于不同账户中，您可以创建 IAM 角色用于跨账户访问。有关更多信息，请参阅 。教程：使用 IAM 角色委托跨 AWS 账户的访问权限。创建跨账户角色之后，为跨账户角色指定 Amazon 资源名称 (ARN) 作为`RoleARN`键/值对。然后，在访问与该接收器的接收器类型关联的 AWS 资源时，Windows Kinesis Agent 尝试代入指定的跨账户角色。
- 如果 Windows Kinesis Agent 在 Amazon EC2 之外执行（例如，本地），则存在多个选项：
  - 如果允许将本地服务器或台式计算机注册为 Amazon EC2 系统管理器托管实例，请使用以下过程来配置身份验证：
    1. 使用在混合环境中设置 AWS Systems Manager 中介绍的过程来设置服务角色、为托管实例创建激活以及安装 SSM 代理。
    2. 将相应的策略附加到服务角色，以允许 Windows Kinesis Agent 访问从所配置接收器流式传输数据时所需的资源。有关将哪些策略附加到 IAM 角色的信息，请参阅配置授权。
    3. 使用配置`ProfileRefreshingAWSCredentialProvider`刷新 AWS 凭证刷新 AWS 凭证。

    由于身份证明由 SSM 和 AWS 安全地管理，因此这是建议用于非 EC2 实例的方法。
  - 如果允许在特定用户而不是默认系统帐户下为 Windows 运行 AWSKinesis sisTAP 服务，请使用以下过程：
    1. 在将使用 AWS 服务的 AWS 账户中创建 IAM 用户。在创建过程中，收集此用户的访问密钥和私有密钥。此过程中稍后的步骤需要这些信息。
    2. 将策略附加到 IAM 用户，授予对所需服务的所需操作访问权限。有关将哪些策略附加到 IAM 用户的信息，请参阅配置授权。
    3. 在各个台式机或服务器上更改 AWSKinesisTap 服务，使其运行在特定用户而不是默认系统账户下。
    4. 使用之前记录的访问密钥和私有密钥在开发工具包存储中创建配置文件。有关更多信息，请参阅配置 AWS 凭证。
    5. 在 `%PROGRAMFILES%\Amazon\AWSKinesisTap` 目录中更新 `AWSKinesisTap.exe.config` 文件，指定在上一步中创建的配置文件的名称。有关更多信息，请参阅配置 AWS 凭证。

对于无法作为托管实例的非 EC2 主机，这是推荐方法，因为特定主机和特定用户的凭证是加密的。

- 如果需要在默认系统账户下为 Windows 运行 AWSKinesis sisTAP 服务，您必须使用共享凭证文件。这是因为系统账户没有用于启用开发工具包存储的 Windows 用户配置文件。共享凭证文件未加密，因此我们建议不要使用此方法。有关如何使用共享配置文件的信息，请参阅配置 AWS 凭证中的适用于 .NET 的 AWS 开发工具包。如果您使用此方法，我们建议您使用 NTFS 加密并限制对共享配置文件的访问权限。管理平台应该轮换密钥，并且在进行密钥轮换时必须更新共享配置文件。

虽然可以在接收器声明中直接提供访问密钥和私有密钥，但不建议使用此方法，因为声明未加密。

配置授权

将相应策略附加到 IAM 用户或角色，适用于 Windows 的 Kinesis Agent 将使用这些用户或角色来流式传输数据到 AWS 服务：

Kinesis Data Streams

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": "arn:aws:kinesis:*:*:stream/*"
        }
    ]
}
```

要将授权限制到特定区域、账户或流名称，请使用特定值替换 ARN 中的相应星号。有关更多信息，请参阅使用 IAM 控制对 Amazon Kinesis Data Streams 资源的访问中的“用于 Kinesis Data Streams 的 Amazon 资源名称 (ARN)”。

Kinesis Data Firehose

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": "arn:aws:firehose:*:*:deliverystream/*"
        }
    ]
}
```

要将授权限制到特定区域、账户或传输流名称，请使用特定值替换 ARN 中的相应星号。有关更多信息，请参阅 。使用 Amazon Kinesis Data Firehose 控制访问中的Amazon Kinesis Data Firehose 开发人员指南。

CloudWatch

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        }
    ]
}
```

有关更多信息，请参阅 。管理您的 CloudWatch 资源的访问权限概述中的Amazon CloudWatch Logs 用户指南。

具有现有日志组和日志流的 CloudWatch 日志

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:*"
        },
        {
            "Sid": "VisualEditor4",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:*:*:log-group:*:*:*"
        }
    ]
}
```

要限制对特定区域、账户、日志组或日志流的访问，请使用相应值替换 ARN 中的相应星号。有关更多信息，请参阅 。管理您的 CloudWatch Logs 资源的访问权限概述中的Amazon CloudWatch Logs 用户指南。

CloudWatch Logs 具有额外权限的 CloudWatch Kinesis 日志日志日志日志，用于 Windows 创建日志组和日志流

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor5",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:*"
        },
```

```
        {
            "Sid": "VisualEditor6",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:*:*:log-group:*:*:*"
        },
        {
            "Sid": "VisualEditor7",
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "*"
        }
    ]
}
```

要限制对特定区域、账户、日志组或日志流的访问，请使用相应值替换 ARN 中的相应星号。有关更多信息，请参阅 。管理您的 CloudWatch Logs 资源的访问权限概述中的Amazon CloudWatch Logs 用户指南。

EC2 标签变量扩展所需的权限

通过 `ec2tag` 变量前缀使用变量扩展需要 `ec2:Describe*` 权限。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Sid": "VisualEditor8",
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    }
   ]
}
```

> **Note**
>
> 只要各个语句的 `Sid` 在策略中唯一，您就可以将多个语句组合成单个策略。有关创建策略的信息，请参阅创建 IAM 策略中的IAM 用户指南。

配置ProfileRefreshingAWSCredentialProvider刷新 AWS 凭证

如果您使用适用于混合环境的 AWS Systems Manager 来管理 AWS 证书，Systems Manager 会在c:\Windows\System32\config\systemprofile\.aws\credentials。有关混合环境的更多信息，请参阅为混合环境设置 AWS Systems Manager中的AWS Systems Manager 用户指南。

由于 AWS .net 开发工具包不会自动获取新证书，因此我们提供ProfileRefreshingAWSCredentialProvider插件刷新凭据。

您可以使用CredentialRef属 AWS 来引用Credentials定义，其中CredentialType属性设置为ProfileRefreshingAWSCredentialProvider如以下示例所示。

```
{
    "Sinks": [{
        "Id": "myCloudWatchLogsSink",
        "SinkType": "CloudWatchLogs",
        "CredentialRef": "ssmcred",
        "Region": "us-west-2",
        "LogGroup": "myLogGroup",
        "LogStream": "myLogStream"
    }],
    "Credentials": [{
        "Id": "ssmcred",
        "CredentialType": "ProfileRefreshingAWSCredentialProvider",
        "Profile": "default",
        "FilePath": "%USERPROFILE%//.aws//credentials",
        "RefreshingInterval": 300
    }]
}
```

凭证定义由以下属性组成，作为键 — 值对。

`Id`

定义接收器定义可以使用CredentialRef以引用此凭据配置。

`CredentialType`

设置为文本字符串ProfileRefreshingAWSCredentialProvider。

`Profile`

可选。默认为 default。

`FilePath`

可选。指定 AWS 凭证文件的路径。如果省略，则 `%USERPROFILE%/.aws/credentials` 是默认值。

`RefreshingInterval`

可选。刷新凭据的频率（以秒为单位）。如果省略，则 `300` 是默认值。

配置接收器修饰

接收器声明可以选择包括指定要流式传输到各种 AWS 服务的额外数据的键/值对，用于增强从源收集的记录。

`TextDecoration`

在接收器声明中未指定 `Format` 时，使用此键/值对。该值是进行变量替换的特殊格式字符串。例如，假设为接收器提供了 `"{ComputerName}:::{timestamp:yyyy-MM-dd HH:mm:ss}:::{_record}"` 的 `TextDecoration`。当源发出包含文本 `The system has resumed from sleep.` 的日志记录时，源通过管道连接到接收器，然后文本 `MyComputer1:::2017-10-26 06:14:22:::The system has resumed from sleep.` 流式传输到与该接收器类型关联的 AWS 服务。`{_record}` 变量引用源所传输的原始文本记录。

`ObjectDecoration`

在接收器声明中指定了 `Format` 时，使用此键/值对，以在记录序列化之前添加其他数据。例如，假设为指定 JSON `Format` 的接收器提供了 `"ComputerName={ComputerName};DT={timestamp:yyyy-MM-dd HH:mm:ss}"` 的 `ObjectDecoration`。流式传输到与接收器类型关联的 AWS 服务的所生成 JSON，在来自源的原始数据之外，包括下列键/值对：

```
{
    ComputerName: "MyComputer2",
    DT: "2017-10-17 21:09:04"
}
```

有关使用 `ObjectDecoration` 的示例，请参阅教程：使用适用于 Windows 的 Kinesis 代理将 JSON 日志文件流式传输到 Amazon S3。

ObjectDecorationEx

指定一个表达式，该表达式允许与ObjectDecoration。此字段可用于当汇的格式为json。表达式语法如以下所示。

```
"ObjectDecorationEx":
 "attribute1={expression1};attribute2={expression2};attribute3={expression3}(;...)"
```

例如，以下ObjectDecorationEx属性：

```
"ObjectDecorationEx":
 "host={env:ComputerName};message={upper(_record)};time={format(_timestamp,
 'yyyyMMdd')}"
```

转换文字记录：

```
System log message
```

使用表达式返回的值进入 JSON 对象，如下所示：

```
{
    "host": "EC2AMAZ-1234",
    "message": "SYSTEM LOG MESSAGE",
    "time": "20210201"
}
```

有关编制表达式的详细信息，请参阅有关编写表达式的提示。其中，ObjectDecoration声明应该使用新语法工作，但时间戳变量除外。A{timestamp:yyyyMMdd}中的字段ObjectDecoration表示为{format(_timestamp,'yyyyMMdd')}在ObjectDecorationEx。

TextDecorationEx

指定一个表达式，该表达式允许与TextDecoration，如以下示例所示。

```
"TextDecorationEx": "Message '{lower(_record)}' at {format(_timestamp, 'yyyy-MM-
dd')}"
```

您可以使用TextDecorationEx来组成 JSON 对象。使用 '@ {'转义左大括号，如以下示例所示。

```
"TextDecorationEx": "@{ \"var\": \"{upper($myvar1)}\" }"
```

如果与接收器连接的源的源类型是 DirectorySource，则接收器可以使用三个其他变量：

_FilePath

日志文件的完整路径。

_FileName

文件名和文件的扩展名。

_Position

一个整数，表示记录在日志文件中的位置。

当您使用从连接到单个接收器的多个文件收集日志记录的源并将所有记录流式传输到单个流时，这些变量非常有用。将这些变量的值注入流记录中可在数据管道的下游进行分析，以按照文件和在各个文件中的位置来排序记录。

有关编写表达式的提示

表达式可以为以下任一种：

- 变量表达式。
- 常量表达式，例如'hello'、1、1.21、null、true、false。
- 调用函数的调用表达式，如以下示例所示。

```
regexp_extract('Info: MID 118667291 ICID 197973259 RID 0 To: <jd@acme.com>', 'To: (\\
\\S+)', 1)
```

特殊字符

需要两个反斜杠来转义特殊字符。

Nesting

函数调用可以嵌套，如以下示例所示。

```
format(date(2018, 11, 28), 'MMddyyyy')
```

Variables

有三种类型的变量：局部变量、元变量和全局变量。

- 局部变量从开始$例如$message。它们用于解析事件对象的属性、如果事件是字典的条目或事件是 JSON 对象的属性。如果局部变量包含空格或特殊字符，请使用引号的局部变量，例如$'date created'。
- 元变量以下划线开头（_），并用于解析为事件的元数据。所有事件类型都支持以下元变量。

  `_timestamp`

  事件的时间戳。

  `_record`

  事件的原始文本表示形式。

  日志事件支持以下附加元变量。

  `_filepath`

  `_filename`

  `_position`

  `_linenumber`

- 全局变量解析为环境变量、EC2 实例元数据或 EC2Tag。为了获得更好的性能，我们建议您使用前缀限制搜索范围，例如`{env:ComputerName}`、`{ec2:InstanceId}`, 和`{ec2tag:Name}`。

内置函数

Windows Kinesis 代理支持以下内置函数。如果任何参数是NULL并且该函数不是为了处理NULL，aNULL对象被返回。

```
//string functions
int length(string input)
string lower(string input)
string lpad(string input, int size, string padstring)
string ltrim(string input)
string rpad(string input, int size, string padstring)
string rtrim(string input)
```

```
string substr(string input, int start)
string substr(string input, int start, int length)
string trim(string input)
string upper(string str)

//regular expression functions
string regexp_extract(string input, string pattern)
string regexp_extract(string input, string pattern, int group)

//date functions
DateTime date(int year, int month, int day)
DateTime date(int year, int month, int day, int hour, int minute, int second)
DateTime date(int year, int month, int day, int hour, int minute, int second, int
 millisecond)

//conversion functions
int? parse_int(string input)
decimal? parse_decimal(string input)
DateTime? parse_date(string input, string format)
string format(object o, string format)

//coalesce functions
object coalesce(object obj1, object obj2)
object coalesce(object obj1, object obj2, object obj3)
object coalesce(object obj1, object obj2, object obj3, object obj4)
object coalesce(object obj1, object obj2, object obj3, object obj4, object obj5)
object coalesce(object obj1, object obj2, object obj3, object obj4, object obj5, object
 obj6)
```

配置接收器变量替换

KinesisStream、KinesisFirehose 和 CloudWatchLogs 接收器声明需要 LogStream 或 StreamName 键/值对。值的值可以包含由 Windows 的 Kinesis 代理自动解析的变量引用。适用于CloudWatchLogs，LogGroup键/值对也是必需的，并且可以包含由 Windows Kinesis 代理自动解析的变量引用。使用模板 {*prefix*:*variablename*} 指定变量，其中 *prefix*: 可选。支持的前缀如下所示：

- env— 变量引用解析为同名的环境变量的值。
- ec2— 变量引用解析为同名的 EC2 实例元数据。
- ec2tag— 变量引用解析为同名的 EC2 实例标签的值。访问实例标签需要 ec2:Describe* 权限。有关更多信息，请参阅 EC2 标签变量扩展所需的权限。

如果未指定前缀，并且某个环境变量具有与 variablename 相同的名称，变量引用解析为环境变量的值。否则，如果 variablename 为 instance_id 或 hostname，变量引用解析为同名的 EC2 元数据的值。如果均不满足，不解析变量引用。

以下为使用变量引用的有效键/值对示例：

```
"LogStream": "LogStream_{instance_id}"
"LogStream": "LogStream_{hostname}"
"LogStream": "LogStream_{ec2:local-hostname}"
"LogStream": "LogStream_{computername}"
"LogStream": "LogStream_{env:computername}"
```

CloudWatchLogs 接收器声明支持特殊格式的时间戳变量，允许来自源的日志或事件记录的时间戳变更日志流的名称。格式为 {timestamp:timeformat}。请参见以下示例：

```
"LogStream": "LogStream_{timestamp:yyyyMMdd}"
```

如果日志或事件记录在 2017 年 6 月 5 日生成，上一例中 LogStream 键/值对的值将解析为 "LogStream_20170605"。

如果已授权，根据生成的名称，在需要时 CloudWatchLogs 接收器类型可以自动创建新日志流。对于其他接收器类型，由于它们在流名称之外需要其他配置，您无法执行此操作。

这些是在文本和对象修饰中出现的特殊变量替换。有关更多信息，请参阅 配置接收器修饰。

配置接收器排队

KinesisStream、KinesisFirehose 和 CloudWatchLogs 接收器声明可以选择如果由于临时连接问题，记录无法流式传输到与这些接收器类型关联的 AWS 服务，则启用记录排队。要启用排队和在连接恢复时的自动流式传输重试，请在接收器声明中使用以下键/值对：

QueueType

指定要使用的排队机制的类型。唯一支持的值是 file，这表示记录应在文件中排队。此键-值对是必需的，以启用适用于 Windows 的 Kinesis 代理的队列功能。如果未指定此项，则默认行为是仅在内存中排队，达到内存排队限制时将无法流式传输。

`QueuePath`

指定包含已排队记录的文件的文件夹。键/值对是可选的。默认值为 `%PROGRAMDATA%\KinesisTap\Queue\SinkId`，其中 SinkId 是您分配作为接收器声明的 `Id` 值的标识符。

`QueueMaxBatches`

限制 Windows Kinesis Agent 在排队记录以进行流式传输时可以使用的空间总量。空间量限制为此键/值对的值乘以每一批的最大字节数。`KinesisStream`、`KinesisFirehose` 和 `CloudWatchLogs` 接收器类型的每一批的最大字节数分别为 5 MB、4 MB 和 1 MB。达到此限制时，任何失败的流式传输不排队，并报告为不可恢复的故障。键/值对是可选的。默认值为 10000 个批处理。

为接收器配置代理

要为访问 AWS 服务的 Windows 接收器类型配置代理，请编辑位于的 Windows Kinesis Agent 配置文件。`%Program Files%\Amazon\KinesisTap\AWSKinesisTap.exe.config`。有关说明，请参阅`proxy`中的部分AWS SDK for .NET 的配置文件参考中的AWS SDK for .NET Developer Guide。

在更多汇属性中配置解析变量

以下示例显示了接收器配置，该接收器配置使用`Region`环境变量的值`Region`属性键/值对。适用于`RoleARN`，它指定 EC2 标签密钥`MyRoleARN`，它的计算结果为与该键关联的值。

```
"Id": "myCloudWatchLogsSink",
"SinkType": "CloudWatchLogs",
"LogGroup": "EC2Logs",
"LogStream": "logs-{instance_id}"
"Region": "{env:Region}"
"RoleARN": "{ec2tag:MyRoleARN}"
```

在 AWS 汇中使用 RoLearn 属性时配置 AWS STS 区域终端节点

此功能仅在您在 Amazon EC2 上使用 KinesiSTAP 并使用`RoleARN`属性代入外部 IAM 角色，以便通过目标 AWS 服务进行身份验证。

通过将设置`UseSTSRegionalEndpoints`到`true`，您可以指定代理使用区域终端节点（例如`https://sts.us-east-1.amazonaws.com`）而不是全局终端节点（例如，`https://sts.amazonaws.com`)。使用区域 STS 终端节点可减少操作的往返延迟，并限制全局终端服务中故障的影响。

为 AWS 汇配置 VPC 终端节点

您可以在接收器配置中指定 VPC 终端节点`CloudWatchLogs`、`CloudWatch`、`KinesisStreams`,和`KinesisFirehose`汇类型。VPC 终端节点使您能够将 VPC 私密地连接到支持的 AWS 服务和 VPC 终端节点服务（由 AWS PrivateLink 提供支持），而无需互联网网关、NAT 设备、VPN 连接或 AWS Direct Connect 连接。VPC 中的实例无需公有 IP 地址便可与服务中的资源通信。VPC 和其他服务之间的通信不会离开 Amazon 网络。有关更多信息，请参阅 。VPC 终端节点中的Amazon VPC 用户指南。

您可 VPC 使用`ServiceURL`属性，如以下示例所示。`CloudWatchLogs`汇配置。将值设置为`ServiceURL`设置为VPC 终端节点节点详选项卡上，使用 Amazon VPC 控制台。

```
{
    "Id": "myCloudWatchLogsSink",
    "SinkType": "CloudWatchLogs",
    "LogGroup": "EC2Logs",
    "LogStream": "logs-{instance_id}",
    "ServiceURL":"https://vpce-ab1c234de56-ab7cdefg.logs.us-east-1.vpce.amazonaws.com"
}
```

配置替代代代理方式

此功能允许您使用 AWS 开发工具包内置的代理支持而不是 .NET 来配置接收器配置代理服务器。以前，将代理配置为使用代理的唯一方法是使用 .NET 的本机功能，该功能通过代理文件中定义的代理自动路由所有 HTTP/S 请求。

如果您当前使用代理与代理服务器一起使用，则无需更改即可使用此方法。

您可以使用`ProxyHost`和`ProxyPort`属性配置备用代理，如以下示例所示。

```
{
    "Id": "myCloudWatchLogsSink",
    "SinkType": "CloudWatchLogs",
    "LogGroup": "EC2Logs",
    "LogStream": "logs-{instance_id}",
    "Region": "us-west-2",
    "ProxyHost": "myproxy.mydnsdomain.com",
    "ProxyPort": "8080"
}
```

管道声明

使用管道声明连接源（请参阅源声明）添加到一个接收器（请参阅接收器声明）在 Amazon Kinesis 代理 Microsoft Windows。管道声明表示为一个 JSON 对象。在启动 Windows 的 Kinesis 代理之后，为指定管道从源收集日志、事件或指标。然后，使用与该管道关联的接收器，将这些数据流式传输到各个 AWS 服务。

下面是示例 管道声明：

```
{
   "Id": "MyAppLogToCloudWatchLogs",
   "SourceRef": "MyAppLog",
   "SinkRef": "MyCloudWatchLogsSink"
}
```

主题

- 配置管道
- 为 Windows 度量管道配置 Kinesis 代理

配置管道

所有管道声明可包含以下键/值对：

`Id`

指定管道的名称（必需）。该名称在配置文件中必须唯一。

`Type`

指定在将日志数据从源传输到接收器时，管道应用的转换类型（如有）。`RegexFilterPipe` 是唯一受支持的值。此值启用对日志记录的基本文本表示进行正则表达式筛选。使用筛选功能可仅将相关日志记录发送到下游数据管道，从而减少传输和存储成本。键/值对是可选的。默认值为不提供转换。

`FilterPattern`

指定在传输到接收器之前，对源所收集日志记录进行筛选所用的 `RegexFilterPipe` 管道的正则表达式。当正则表达式与记录的基本文本表示匹配时，由 `RegexFilterPipe` 类型管道传输日志记录。所生成的结构化日志记录（例如，在 `DirectorySource` 声明中使用 `ExtractionPattern` 键/值对时），仍可使用 `RegexFilterPipe` 机制来筛选。这是因为此机制

在解析之前对原始文本表示进行操作。此键/值对是可选的，但在管道指定 RegexFilterPipe 类型时必须提供。

下面是示例 RegexFilterPipe 管道声明：

```
{
 "Id": "MyAppLog2ToFirehose",
 "Type": "RegexFilterPipe",
 "SourceRef": "MyAppLog2",
 "SinkRef": "MyFirehose",
 "FilterPattern": "^(10|11),.*",
 "IgnoreCase": false,
 "Negate": false
}
```

SourceRef

指定源声明的名称（Id 键/值对的值），该声明定义为管道收集日志、事件和指标的源（必需）。

SinkRef

指定接收器声明的名称（Id 键/值对的值），该声明定义接收管道的日志、事件和指标的接收器（必需）。

IgnoreCase

可选。接受的值true或者false。如果设置为true，则正则表达式将以不区分大小写的方式匹配记录。

Negate

可选。接受的值true或者false。如果设置为true，管道将转发不匹配正则表达式。

有关使用 RegexFilterPipe 管道类型的完整配置文件示例，请参阅使用管道。

为 Windows 度量管道配置 Kinesis 代理

名为的内置指标源_KinesisTapMetricsSource，生成有关适用于 Windows 的 Kinesis 代理的衡量指标。如果有CloudWatch接收器声明Id的MyCloudWatchSink，以下示例管道声明将 Windows 生成指标的 Kinesis 代理传输到该接收器：

```
{
```

```
    "Id": "KinesisAgentMetricsToCloudWatch",
    "SourceRef": "_KinesisTapMetricsSource",
    "SinkRef": "MyCloudWatchSink"
}
```

有关 Windows 的 Kinesis 代理内置指标源的更多信息，请参阅Windows 内置指标源的 Kinesis 代理。

如果配置文件还流式传输 Windows 性能计数器指标，我们建议您使用单独的管道和接收器，而不是为 Windows 指标和 Windows 性能计数器指标使用相同的接收器。

配置自动更新

使用appsettings.json配置文件，以启用 Microsoft Windows 的 Amazon Kinesis 代理的自动更新以及适用于 Windows 的 Kinesis 代理的配置文件。要控制更新行为，请在配置文件中与 Sources、Sinks 和 Pipes 相同的级别指定 Plugins 键/值对。

Plugins 键/值对指定不在源、接收器和管道类别中时使用的其他一般功能。例如，有一个插件用于更新 Windows 的 Kinesis 代理，还有一个插件用于更新appsettings.json配置文件。插件表示为 JSON 对象，并始终具有 Type 键/值对。Type 确定可以为插件指定的其他键/值对。支持以下插件类型：

PackageUpdate

指定 Windows 的 Kinesis 代理应定期检查程序包版本配置文件。如果程序包版本文件指示应安装不同版本的 Windows Kinesis 代理，则 Windows 的 Kinesis 代理下载该版本并安装。PackageUpdate 插件键/值对包括：

Type

该值必须为字符串 PackageUpdate，并且是必需的。

Interval

指定检查程序包版本文件中是否有任何更改的频率，以分钟为单位并表示为字符串。键/值对是可选的。如果未指定，则默认值为 60 分钟。如果值小于 1，则不进行更新检查。

PackageVersion

指定程序包版本 JSON 文件的位置。该文件可驻留在文件共享 (file://)、一个网站 (http://) 或 Amazon S3 (s3://)。例如，值变为s3://mycompany/config/agent-package-version.json指示 Windows Kinesis 代理应检查config/agent-package-version.json中的文件mycompanyAmazon S3 存储桶。它应根据该文件的内容执行更新。

> **Note**
>
> 的值`PackageVersion`对于 Amazon S3，键/值对对的值区分大小写。

以下是程序包版本文件内容的示例：

```
{
    "Name": "AWSKinesisTap",
    "Version": "1.0.0.106",
    "PackageUrl": "https://s3-us-west-2.amazonaws.com/kinesis-agent-windows/
downloads/AWSKinesisTap.{Version}.nupkg"
}
```

这些区域有：`Version`键/值对指定应安装的 Windows Kinesis 代理版本（如果尚未安装）。`PackageUrl` 中引用的 `{Version}` 变量解析您为 `Version` 键/值对指定的值。在本示例中，变量解析为字符串 `1.0.0.106`。提供此变量解析是为了在程序包版本文件中有一个单独的位置可以用于存储特定所需版本。您可以使用多个程序包版本文件来控制部署新版本的 Windows Kinesis 代理的节奏，以在较大规模的部署之前进行验证。要回退 Windows 的 Kinesis 代理的部署，请更改一个或多个程序包版本文件，指定已知可在您环境中使用的 Windows 的较早版本的 Kinesis 代理。

`PackageVersion` 键/值对的值受变量替换的影响，用于帮助自动选择不同的程序包版本文件。有关变量替换的更多信息，请参阅配置接收器变量替换。

`AccessKey`

指定在对 Amazon S3 中程序包版本文件的访问进行身份验证时，要使用的访问密钥。键/值对是可选的。我们不建议使用此键/值对。有关推荐的替代身份验证过程，请参阅配置身份验证。

`SecretKey`

指定在对 Amazon S3 中程序包版本文件的访问进行身份验证时，要使用的私有密钥。键/值对是可选的。我们不建议使用此键/值对。有关推荐的替代身份验证过程，请参阅配置身份验证。

`Region`

指定在从 Amazon S3 访问程序包版本文件时要使用的区域终端节点。键/值对是可选的。

`ProfileName`

指定在对 Amazon S3 中程序包版本文件的访问进行身份验证时，要使用的安全配置文件。有关更多信息，请参阅 配置身份验证。键/值对是可选的。

`RoleARN`

指定在跨账户场景下，在 Amazon S3 中程序包版本文件的访问进行身份验证时，要代入的角色。有关更多信息，请参阅 配置身份验证。键/值对是可选的。

如果未指定 `PackageUpdate` 插件，则不检查程序包版本文件来确定是否需要更新。

`ConfigUpdate`

指定 Windows Kinesis 代理应定期检查更新的`appsettings.json`配置文件存储在文件共享、网站或 Amazon S3 中。如果存在更新的配置文件，则下载并安装该文件的 Kinesis 代理适用于 Windows。`ConfigUpdate`键/值对包括以下内容：

`Type`

该值必须为字符串 `ConfigUpdate`，并且是必需的。

`Interval`

指定检查新配置文件的频率，以分钟为单位并表示为字符串。此键/值对可选，如果未指定，则默认为 5 分钟。如果值小于 1，则不检查配置文件更新。

`Source`

指定在何处查找已更新的配置文件。该文件可驻留在文件共享 (`file://`)、一个网站 (`http://`) 或 Amazon S3 (`s3://`)。例如，值变为`s3://mycompany/config/appsettings.json`指示 Windows Kinesis 代理应检查`config/appsettings.json`中的文件`mycompany`Amazon S3 存储桶。

> **Note**
>
> 的值`Source`对于 Amazon S3，键/值对的值区分大小写。

`Source` 键/值对的值受变量替换的影响，可用于帮助自动选择不同的配置文件。有关变量替换的更多信息，请参阅配置接收器变量替换。

`Destination`

指定在本地计算机上存储配置文件的位置。这可以是相对路径、绝对路径或者包含环境变量引用的路径，例如 `%PROGRAMDATA%`。如果是相对路径，则它相对于 Windows 的 Kinesis 代理安装的位置。通常值应为 `.\appsettings.json`。此键/值对是必需的。

`AccessKey`

指定在对 Amazon S3 中配置文件的访问进行身份验证时，要使用的访问密钥。键/值对是可选的。我们不建议使用此键/值对。有关推荐的替代身份验证过程，请参阅配置身份验证。

`SecretKey`

指定在对 Amazon S3 中配置文件的访问进行身份验证时，要使用的私有密钥。键/值对是可选的。我们不建议使用此键/值对。有关推荐的替代身份验证过程，请参阅配置身份验证。

`Region`

指定在从 Amazon S3 访问配置文件时要使用的区域终端节点。键/值对是可选的。

`ProfileName`

指定在对 Amazon S3 中配置文件的访问进行身份验证时，要使用的安全配置文件。有关更多信息，请参阅 配置身份验证。键/值对是可选的。

`RoleARN`

指定在跨账户场景下，在对 Amazon S3 中配置文件的访问进行身份验证时，要代入的角色。有关更多信息，请参阅 配置身份验证。键/值对是可选的。

如果未指定 `ConfigUpdate` 插件，则不检查配置文件来确定是否需要进行配置文件更新。

下面是一个示例 `appsettings.json` 配置文件，该文件演示 `PackageUpdate` 和 `ConfigUpdate` 插件的使用。在本示例中，程序包版本文件位于`mycompany`Amazon S3 存储桶名为`config/agent-package-version.json`。大约每 2 个小时检查此文件是否有任何更新。如果在程序包版本文件中指定了不同版本的 Kinesis Agent，则从程序包版本文件中的指定位置安装指定的代理版本。

此外，还有一个`appsettings.json`配置文件存储在`mycompany`Amazon S3 存储桶名为`config/appsettings.json`。大约每 30 分钟将该文件与当前配置文件进行对比。如果发生更改，则从 Amazon S3 下载更新的配置文件并安装到`appsettings.json`配置文件。

```
{
  "Sources": [
    {
      "Id": "ApplicationLogSource",
      "SourceType": "DirectorySource",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "RecordParser": "SingleLine"
    }
```

```
  ],
  "Sinks": [
    {
       "Id": "ApplicationLogKinesisFirehoseSink",
       "SinkType": "KinesisFirehose",
       "StreamName": "ApplicationLogFirehoseDeliveryStream",
       "Region": "us-east-1"
    }
    ],
  "Pipes": [
    {
      "Id": "ApplicationLogSourceToApplicationLogKinesisFirehoseSink",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "ApplicationLogKinesisFirehoseSink"
    }
  ],
  "Plugins": [
    {
      "Type": "PackageUpdate"
      "Interval": "120",
      "PackageVersion": "s3://mycompany/config/agent-package-version.json"
    },
    {
      "Type": "ConfigUpdate",
      "Interval": "30",
      "Source": "s3://mycompany/config/appsettings.json",
      "Destination": ".\appSettings.json"
    }
  ]
}
```

适用于 Windows 的 Kinesis 代理配置示例

这些区域有：appsettings.json配置文件是一个 JSON 文档，控制面向 Microsoft Windows 的 Amazon Kinesis 代理如何收集日志、事件和指标。它还控制 Windows Kinesis Agent 如何将数据转换为数据并将其流式传输到各个 AWS 服务。有关配置文件中的源、接收器和管道声明的详细信息，请参阅源声明、接收器声明和管道声明。

以下部分包含多个不同类型场景的配置文件示例。

主题

- 从不同源流式传输到 Kinesis Data Streams

- 从 Windows 应用程序事件日志流式传输到接收器
- 使用管道
- 使用多个源和管道

从不同源流式传输到 Kinesis Data Streams

以下示例`appsettings.json`配置文件演示将来自不同源的日志和事件流式传输到 Kinesis Data Streams 指标，以及从 Windows 性能计数器流式传输到 Amazon CloudWatch 指标。

`DirectorySource`、`SysLog` 记录解析程序

以下文件将 syslog 格式的日志记录，从所有文件，流式传输到`.log`文件扩展名`C:\LogSource\`目录中的`SyslogKinesisDataStream`us-east-1 区域中的 Kinesis Data Streams。其中将建立一个书签，确保发送来自日志文件的所有数据，即使代理关闭并稍后重启。自定义应用程序可以读取和处理来自 `SyslogKinesisDataStream` 流的记录。

```
{
  "Sources": [
    {
      "Id": "SyslogDirectorySource",
      "SourceType": "DirectorySource",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "RecordParser": "SysLog",
      "TimeZoneKind": "UTC",
      "InitialPosition": "Bookmark"
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "SyslogKinesisDataStream",
      "Region": "us-east-1"
    }
  ],
  "Pipes": [
    {
      "Id": "SyslogDS2KSSink",
      "SourceRef": "SyslogDirectorySource",
```

```
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

DirectorySource、SingleLineJson 记录解析程序

以下文件将 JSON 格式的日志记录，从所有文件，流式传输到.log文件扩展名C:\LogSource\目录中的JsonKinesisDataStreamus-east-1 区域中的 Kinesis Data Streams。在流式传输之前，ComputerName 的键/值对和 DT 键添加到各个 JSON 对象中，并带有计算机名称的值以及处理记录的日期和时间。自定义应用程序可以读取和处理来自 JsonKinesisDataStream 流的记录。

```
{
  "Sources": [
    {
      "Id": "JsonLogSource",
      "SourceType": "DirectorySource",
      "RecordParser": "SingleLineJson",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "InitialPosition": 0
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "JsonKinesisDataStream",
      "Region": "us-east-1",
      "Format": "json",
      "ObjectDecoration": "ComputerName={ComputerName};DT={timestamp:yyyy-MM-dd
 HH:mm:ss}"
    }
  ],
  "Pipes": [
    {
      "Id": "JsonLogSourceToKinesisStreamSink",
      "SourceRef": "JsonLogSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

ExchangeLogSource

以下文件将 Microsoft Exchange 生成的日志记录，以及存储在.log扩展名C:\temp\ExchangeLog\目录中的ExchangeKinesisDataStream以 JSON 格式显示的 us-east-1 区域中的 Kinesis 力学数据流。虽然 Exchange 日志并非 JSON 格式，Windows Kinesis 代理程序可以解析日志并将其传输到 JSON。在流式传输之前，ComputerName 的键/值对和 DT 键添加到各个 JSON 对象中，其中包含计算机名称的值以及处理记录的日期和时间。自定义应用程序可以读取和处理来自 ExchangeKinesisDataStream 流的记录。

```
{
  "Sources": [
    {
       "Id": "ExchangeSource",
       "SourceType": "ExchangeLogSource",
       "Directory": "C:\\temp\\ExchangeLog\",
       "FileNameFilter": "*.log"
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "ExchangeKinesisDataStream",
      "Region": "us-east-1",
      "Format": "json",
      "ObjectDecoration": "ComputerName={ComputerName};DT={timestamp:yyyy-MM-dd
 HH:mm:ss}"
    }
  ],
  "Pipes": [
    {
      "Id": "ExchangeSourceToKinesisStreamSink",
      "SourceRef": "ExchangeSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

W3SVCLogSource

以下文件将存储在这些文件标准位置的 Windows 日志记录，流式传输到IISKinesisDataStreamus-east-1 区域中的 Kinesis Data Streams。自定义应用程序可以读取和处理来自 IISKinesisDataStream 流的记录。IIS 是适用于 Windows 的 Web 服务器。

```
{
  "Sources": [
    {
       "Id": "IISLogSource",
       "SourceType": "W3SVCLogSource",
       "Directory": "C:\\inetpub\\logs\\LogFiles\\W3SVC1",
       "FileNameFilter": "*.log"
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "IISKinesisDataStream",
      "Region": "us-east-1"
    }
  ],
  "Pipes": [
    {
      "Id": "IISLogSourceToKinesisStreamSink",
      "SourceRef": "IISLogSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

带有查询的 WindowsEventLogSource

以下文件将日志事件从 Windows 系统事件日志进行流式处理，这些事件具有Critical或者Error（小于或等于 2）设置为SystemKinesisDataStream以 JSON 格式显示的 us-east-1 区域中的 Kinesis 力学数据流。自定义应用程序可以读取和处理来自 SystemKinesisDataStream 流的记录。

```
{
  "Sources": [
    {
```

```
        "Id": "SystemLogSource",
        "SourceType": "WindowsEventLogSource",
        "LogName": "System",
        "Query": "*[System/Level<=2]"
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "SystemKinesisDataStream",
      "Region": "us-east-1",
      "Format": "json"
    }
  ],
  "Pipes": [
    {
      "Id": "SLSourceToKSSink",
      "SourceRef": "SystemLogSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

WindowsETWEventSource

以下文件将 Microsoft 公共语言运行时 (CLR) 异常和安全事件流式传输到ClrKinesisDataStream以 JSON 格式显示的 us-east-1 区域中的 Kinesis 力学数据流。自定义应用程序可以读取和处理来自 ClrKinesisDataStream 流的记录。

```
{
  "Sources": [
    {
       "Id": "ClrETWEventSource",
       "SourceType": "WindowsETWEventSource",
       "ProviderName": "Microsoft-Windows-DotNETRuntime",
       "TraceLevel": "Verbose",
       "MatchAnyKeyword": "0x00008000, 0x00000400"
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
```

```
      "SinkType": "KinesisStream",
      "StreamName": "ClrKinesisDataStream",
      "Region": "us-east-1",
      "Format": "json"
    }
  ],
  "Pipes": [
    {
      "Id": "ETWSourceToKSSink",
      "SourceRef": "ClrETWEventSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

WindowsPerformanceCounterSource

以下文件将所有打开文件的性能计数器、重启以来的登录尝试总数、每秒磁盘读取数以及空闲磁盘空间百分比，流式传输到 us-east-1 区域中的 CloudWatch 指标。您可以在 CloudWatch 中绘制这些指标的图形、从图形构建控制面板以及设置在超过阈值时发送通知的警报。

```
{
  "Sources": [
    {
      "Id": "PerformanceCounter",
      "SourceType": "WindowsPerformanceCounterSource",
      "Categories": [
        {
          "Category": "Server",
          "Counters": [
            "Files Open",
            "Logon Total"
          ]
        },
        {
          "Category": "LogicalDisk",
          "Instances": "*",
          "Counters": [
            "% Free Space",
            {
              "Counter": "Disk Reads/sec",
              "Unit": "Count/Second"
            }
```

```
          ]
        }
      ],
    }
  ],
  "Sinks": [
    {
      "Namespace": "MyServiceMetrics",
      "Region": "us-east-1",
      "Id": "CloudWatchSink",
      "SinkType": "CloudWatch"
    }
  ],
  "Pipes": [
    {
      "Id": "PerformanceCounterToCloudWatch",
      "SourceRef": "PerformanceCounter",
      "SinkRef": "CloudWatchSink"
    }
  ]
}
```

从 Windows 应用程序事件日志流式传输到接收器

以下示例`appsettings.json`配置文件演示将 Windows 应用程序事件日志流式传输到亚 Amazon Kinesis 代理中的各个接收器。有关使用 `KinesisStream` 和 `CloudWatch` 接收器类型的示例，请参阅从不同源流式传输到 Kinesis Data Streams。

`KinesisFirehose`

以下文件流`Critical`或者`Error`Windows 应用程序将事件记录到`WindowsLogFirehoseDeliveryStream`在 us-east-1 区域中的 Kinesis Data Firehose 传输流。如果与 Kinesis Data Firehose 的连接中断，则首先将事件在内存中排队。接下来，如有必要，这些事件在磁盘上的文件中排队，直至恢复连接。然后，事件将出队并发送，后跟任何新事件。

您可以配置 Kinesis Data Firehose，以根据数据管道要求，将流式传输的数据存储到多个不同类型的存储和分析服务。

```
{
  "Sources": [
    {
         "Id": "ApplicationLogSource",
```

```
          "SourceType": "WindowsEventLogSource",
          "LogName": "Application",
          "Query": "*[System/Level<=2]"
      }
    ],
    "Sinks": [
      {
         "Id": "WindowsLogKinesisFirehoseSink",
         "SinkType": "KinesisFirehose",
         "StreamName": "WindowsLogFirehoseDeliveryStream",
         "Region": "us-east-1",
         "QueueType": "file"
      }
      ],
    "Pipes": [
      {
        "Id": "ALSource2ALKFSink",
        "SourceRef": "ApplicationLogSource",
        "SinkRef": "WindowsLogKinesisFirehoseSink"
      }
    ]
}
```

CloudWatchLogs

以下文件流Critical或者ErrorWindows 应用程序日志事件 CloudWatch Logs 事件流式传输到MyServiceApplicationLog-Group日志组。各个流的名称以 Stream- 开头。它以创建流的四位数年份、两位数月份以及两位数日期结尾，所有数字连在一起（例如，Stream-20180501 是创建于 2018 年 5 月 1 日的流）。

```
{
  "Sources": [
    {
          "Id": "ApplicationLogSource",
          "SourceType": "WindowsEventLogSource",
          "LogName": "Application",
          "Query": "*[System/Level<=2]"
    }
  ],
  "Sinks": [
    {
      "Id": "CloudWatchLogsSink",
      "SinkType": "CloudWatchLogs",
```

```
      "LogGroup": "MyServiceApplicationLog-Group",
      "LogStream": "Stream-{timestamp:yyyyMMdd}",
      "Region": "us-east-1",
      "Format": "json"
    }
  ],
  "Pipes": [
    {
      "Id": "ALSource2CWLSink",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "CloudWatchLogsSink"
    }
  ]
}
```

使用管道

以下示例 appsettings.json 配置文件演示使用与管道相关的功能。

此示例将日志条目流式传输到c:\LogSource\添加到ApplicationLogFirehoseDeliveryStreamKinesis Data Firehose 传输流。它仅包含与 FilterPattern 键/值对所指定的正则表达式匹配的行。具体来说，只有日志文件中以10或者11将流式传输到 Kinesis Data Firehose。

```
{
  "Sources": [
    {
      "Id": "ApplicationLogSource",
      "SourceType": "DirectorySource",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "RecordParser": "SingleLine"
    }
  ],
  "Sinks": [
    {
       "Id": "ApplicationLogKinesisFirehoseSink",
       "SinkType": "KinesisFirehose",
       "StreamName": "ApplicationLogFirehoseDeliveryStream",
       "Region": "us-east-1"
    }
    ],
  "Pipes": [
```

```
    {
      "Id": "ALSourceToALKFSink",
      "Type": "RegexFilterPipe",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "ApplicationLogKinesisFirehoseSink",
      "FilterPattern": "^(10|11),.*"
    }
  ]
}
```

使用多个源和管道

以下示例 appsettings.json 配置文件演示使用多个源和管道。

此示例将应用程序、安全和系统 Windows 事件日志流式传输到EventLogStreamKinesis Data Firehose 传输流，使用三个源，三个管道，和一个接收器。

```
{
    "Sources": [
  {
    "Id": "ApplicationLog",
    "SourceType": "WindowsEventLogSource",
    "LogName": "Application"
  },
  {
    "Id": "SecurityLog",
    "SourceType": "WindowsEventLogSource",
    "LogName": "Security"
  },
  {
    "Id": "SystemLog",
    "SourceType": "WindowsEventLogSource",
    "LogName": "System"
  }
    ],
    "Sinks": [
  {
    "Id": "EventLogSink",
    "SinkType": "KinesisFirehose",
    "StreamName": "EventLogStream",
    "Format": "json"
  },
    ],
```

```
    "Pipes": [
  {
    "Id": "ApplicationLogToFirehose",
    "SourceRef": "ApplicationLog",
    "SinkRef": "EventLogSink"
  },
  {
    "Id": "SecurityLogToFirehose",
    "SourceRef": "SecurityLog",
    "SinkRef": "EventLogSink"
  },
  {
    "Id": "SystemLogToFirehose",
    "SourceRef": "SystemLog",
    "SinkRef": "EventLogSink"
  }
    ]
}
```

配置遥测

为实现更好的支持，在默认情况下，适用于微软 Windows 的 Amazon Kinesis 代理收集有关代理的操作并将其发送到 AWS。此信息不包含任何个人可识别信息。它不包含任何您收集或流式传输到 AWS 服务的数据。我们每 60 分钟收集大约 1 到 2 KB 的此指标数据。

您可以选择退出收集和传输这些统计信息。要执行此操作，请将以下的键/值对添加到 `appsettings.json` 配置文件中与源、接收器和管道相同的级别。

```
"Telemetrics":
    { "off": "true" }
```

例如，以下配置文件配置源、接收器和管道，并禁用遥测：

```
{
  "Sources": [
    {
      "Id": "ApplicationLogSource",
      "SourceType": "DirectorySource",
      "Directory": "C:\\LogSource\\",
```

```
      "FileNameFilter": "*.log",
      "RecordParser": "SingleLine"
    }
  ],
  "Sinks": [
    {
       "Id": "ApplicationLogKinesisFirehoseSink",
       "SinkType": "KinesisFirehose",
       "StreamName": "ApplicationLogFirehoseDeliveryStream",
       "Region": "us-east-1"
    }
    ],
  "Pipes": [
    {
      "Id": "ApplicationLogSourceToApplicationLogKinesisFirehoseSink",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "ApplicationLogKinesisFirehoseSink"
    }
  ],
  "Telemetrics":
    {
      "off": "true"
    }
}
```

我们在启用了遥测时收集以下指标：

`ClientId`

安装软件时自动分配的唯一 ID。

`ClientTimestamp`

收集遥测的日期和时间。

`OSDescription`

操作系统的描述。

`DotnetFramework`

当前的 .NET 框架版本。

`MemoryUsage`

用于 Windows 的 Kinesis 代理使用的内存量 (MB)。

`CPUUsage`

以小数表示的 Windows CPU 利用率百分比。例如，0.01 表示 1%。

`InstanceId`

如果适用于 Windows 的 Kinesis 代理正在 Amazon EC2 实例上运行，则会显示 Amazon EC2 实例 ID。

`InstanceType (string)`

Amazon EC2 实例类型（如果 Windows Kinesis 代理程序在 Amazon EC2 实例上运行）。

此外，我们收集在适用于 Windows 指标的 Kinesis 代理列表中列出的指标。

教程：使用适用于 Windows 的 Kinesis 代理将 JSON 日志文件流式传输到 Amazon S3

本教程介绍了使用 Amazon Kinesis Agent Microsoft Windows (Kinesis 代理) 设置数据管道的详细步骤。

本教程包括以下步骤：

- 使用 Windows 版本代理将 JSON Kinesis 式的日志文件流式传输到Amazon Simple Storage Service (Amazon S3)通过Amazon Kinesis Data Firehose。有关适用于 Windows 的 Kinesis 代理的信息，请参阅什么是 Amazon Kinesis Analytics？。
- 在流式传输之前，使用对象修饰增强日志数据。有关更多信息，请参阅 配置接收器修饰。
- 使用Amazon Athena搜索特定类型的日志记录。

Prerequisites

如果您还没有 AWS 账户，请按照设置 AWS 账户来获得一个。

主题

步骤 1: 配置 AWS 服务

执行以下步骤来准备环境，以便使用 Amazon Kinesis Agent 将日志数据流式传输到 Amazon Simple Storage Service (Amazon S3)。有关更多信息和先决条件，请参阅教程：将 JSON 日志文件流式传输到 Amazon S3。

使用 AWS 管理控制台配置 AWS Identity and Access Management (IAM)、Amazon S3、Kinesis Data Firehose 件和亚马逊 Elastic Compute Cloud (Amazon EC2)，以便准备将日志数据从 EC2 实例流式传输到 Amazon S3。

主题

配置 IAM 策略和角色

创建以下策略，该策略授权 Windows Kinesis Agent 将记录流式传输到特定的 Kinesis Data Firehose 传输流：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": "arn:aws:firehose:region:account-id:deliverystream/log-
delivery-stream"
        }
    ]
}
```

Replace*region*，名为 AWS 区域名称，该区域将在其中创建 Kinesis Data Firehose 传输流（`us-east-1`，例如）。将 *account-id* 替换为将在其中创建传输流的 AWS 账户的 12 位账户 ID。

在导航栏中，选择支持，然后支持中心。您当前登录的 12 位数字账户号 (ID) 将显示在支持中心导航窗格中。

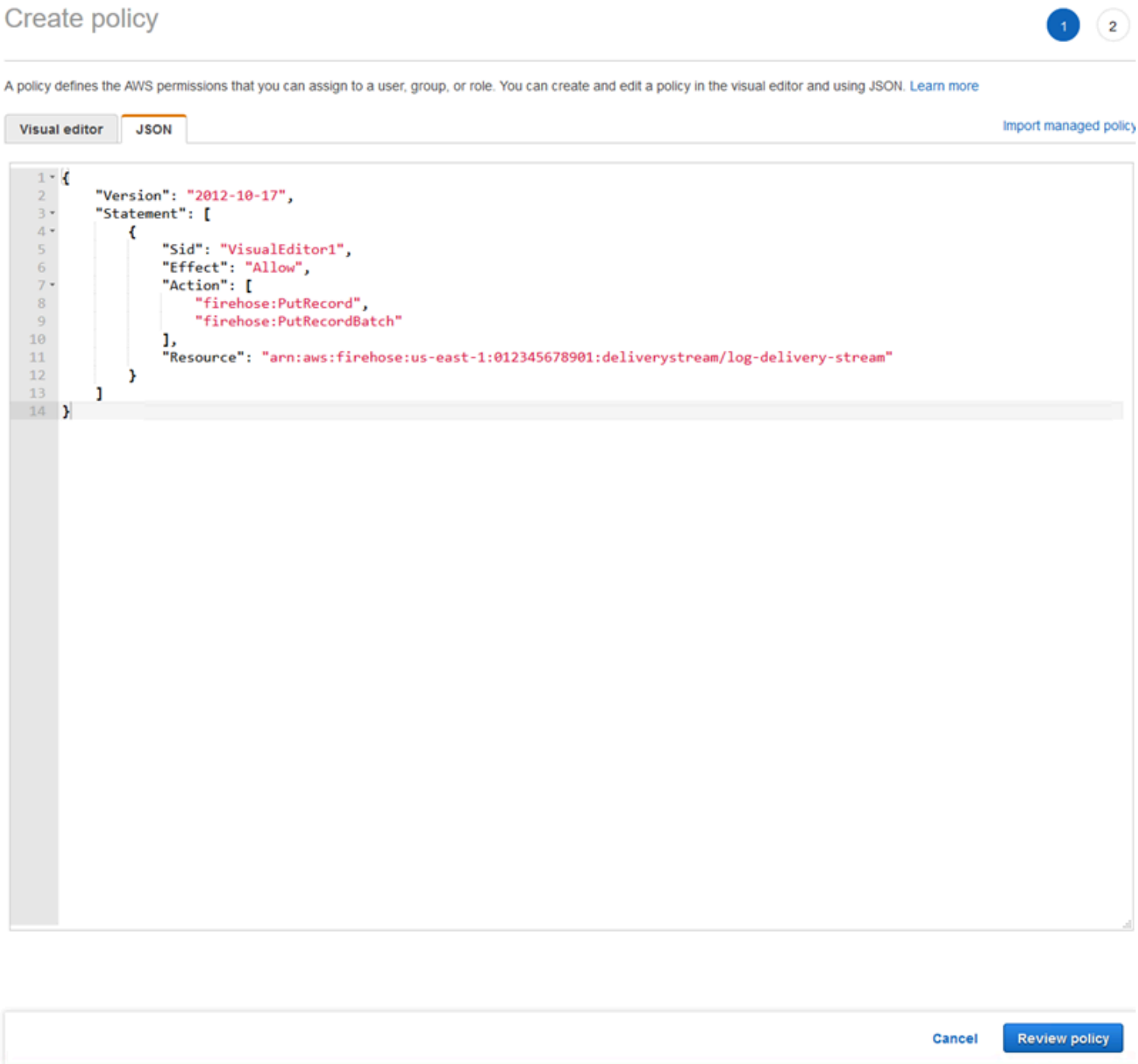
使用以下过程创建策略。将策略命名为 `log-delivery-stream-access-policy`。

使用 JSON 策略编辑器创建策略

1. 登录 AWS 管理控制台，并通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择 Policies (策略)。

   如果这是您首次选择 Policies，则会显示 Welcome to Managed Policies 页面。选择 Get Started。
3. 在页面的顶部，选择 Create policy (创建策略)。
4. 选择 JSON 选项卡。
5. 输入 JSON 策略文档。有关 IAM 策略语言的详细信息，请参阅IAM JSON 策略参考中的IAM 用户指南。
6. 完成后，选择查看策略。策略验证程序将报告任何语法错误。

   > **Note**
   >
   > 您可以随时在可视化编辑器和 JSON 选项卡之间切换。但是，如果您进行更改或选择查看策略中的可视化编辑器选项卡中，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅 。调整策略结构中的IAM 用户指南。
7. 在 Review policy (查看策略) 页面上，为创建的策略输入 Name (名称) 和 Description (描述) (可选) 。查看策略摘要以查看您的策略授予的权限。然后，选择创建策略以保存您的工作。



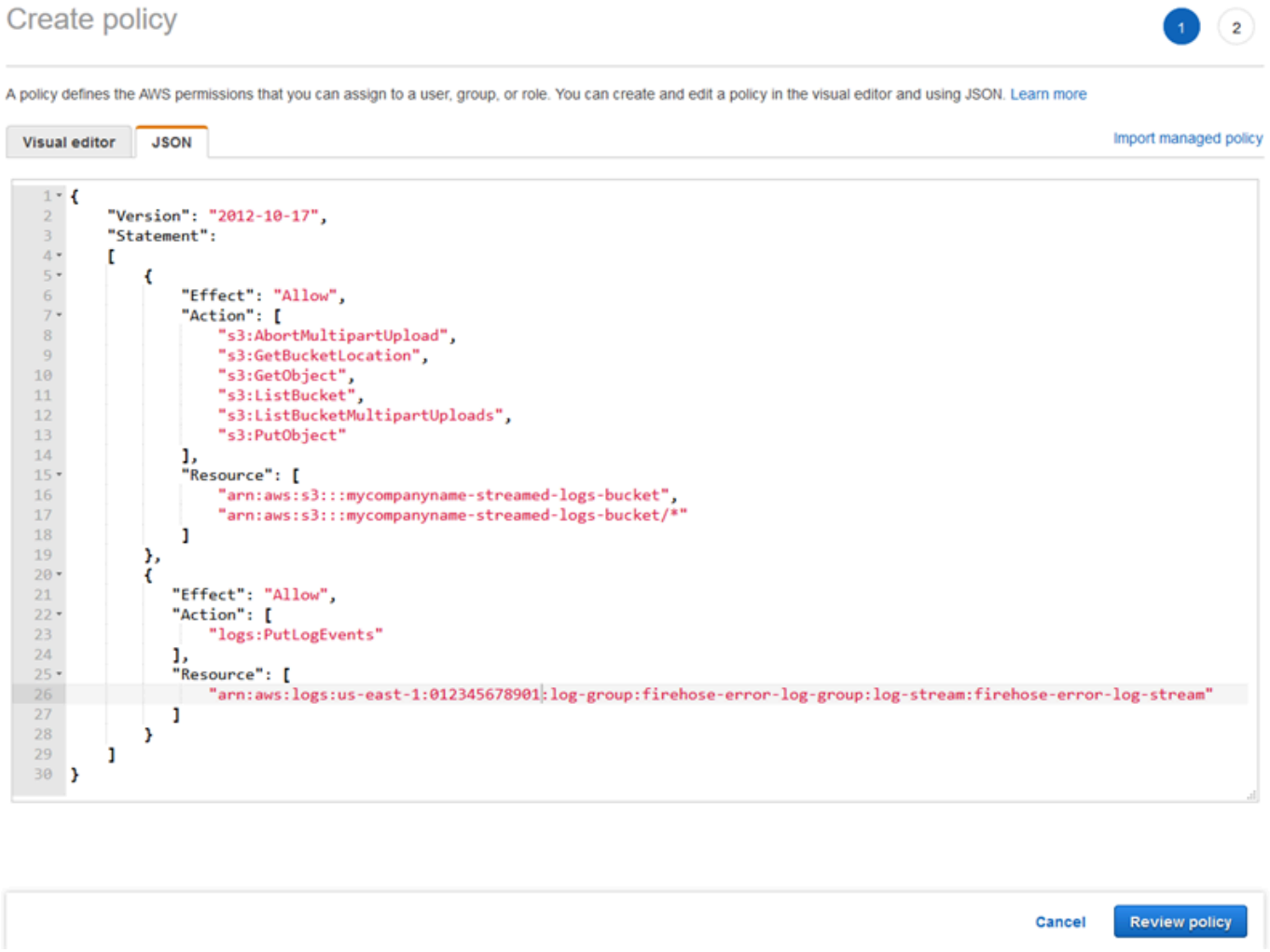
创建向 Kinesis Data Firehose 授予对 S3 存储桶的访问权限的角色

1. 通过使用上一过程，创建一个使用以下 JSON 定义的名为 `firehose-s3-access-policy` 的策略：

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

```
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "logs:PutLogEvents"
           ],
           "Resource": [
               "arn:aws:logs:region:account-id:log-group:firehose-error-log-
group:log-stream:firehose-error-log-stream"
           ]
        }
    ]
}
```

将 *bucket-name* 替换为将存储日志的唯一存储桶名称。Replace*region*（将在其中创建 CloudWatch Logs 组和日志流的 AWS 区域）。它们用于记录在通过 Kinesis Data Firehose 将数据流式传输到 Amazon S3 时发生的任何错误。将 *account-id* 替换为将在其中创建日志组和日志流的账户的 12 位账户 ID。



2. 在 IAM 控制台的导航窗格中，选择 Roles，然后选择 Create role。
3. 选择AWS 服务角色类型，然后选择Kinesis服务。
4. 选择Kinesis Data Firehose作为用例，然后选择后续：权限。
5. 在搜索框中，输入**`firehose-s3-access-policy`**，选择该策略，然后选择后续：审核。
6. 在 Role name (角色名称) 框中，输入 **`firehose-s3-access-role`**。
7. 选择创建角色。

创建要与将运行 Kinesis Agent 的 EC2 实例的实例配置文件关联的角色

1. 在 IAM 控制台的导航窗格中，选择 Roles，然后选择 Create role。
2. 选择AWS 服务角色类型，然后选择EC2。
3. 选择后续：权限。
4. 在搜索框中，输入 **`log-delivery-stream-access-policy`**。

5. 选择策略，然后选择后续：审核。
6. 在 Role name (角色名称) 框中，输入 **kinesis-agent-instance-role**。
7. 选择创建角色。

创建 Amazon S3 存储桶

创建 Kinesis Data Firehose 在其中流式传输日志的 S3 存储桶。

创建用于日志存储的 S3 存储桶

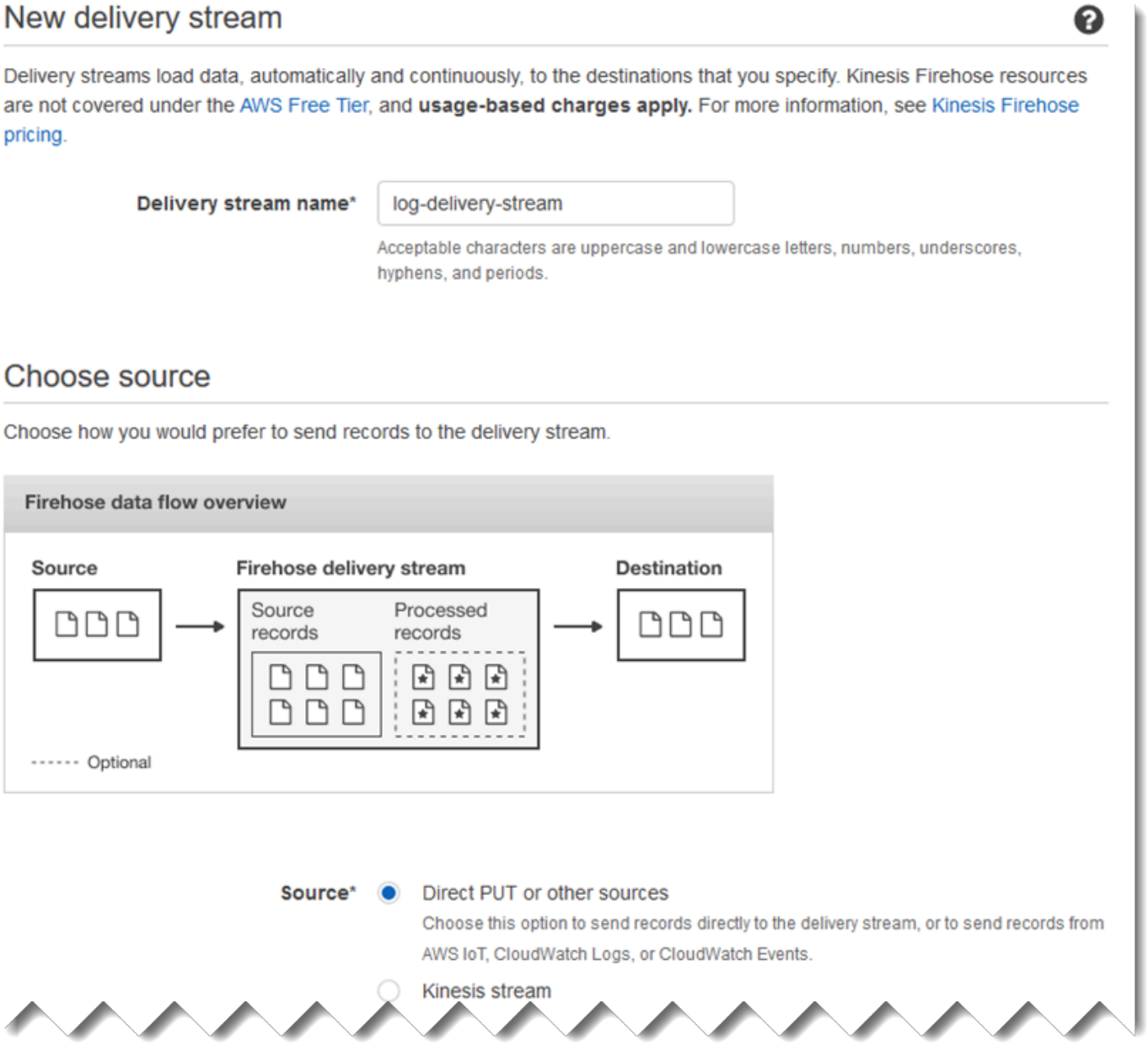
1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 选择 Create bucket (创建存储桶)。
3. 在 Bucket name (存储桶名称) 框中，输入您在配置 IAM 策略和角色中选择的唯一 S3 存储桶名称。
4. 选择应创建存储桶的区域。这通常与您打算在其中创建 Kinesis Data Firehose 传输流和 Amazon EC2 实例的区域相同。
5. 选择创建。

创建 Kinesis Data Firehose 传输流

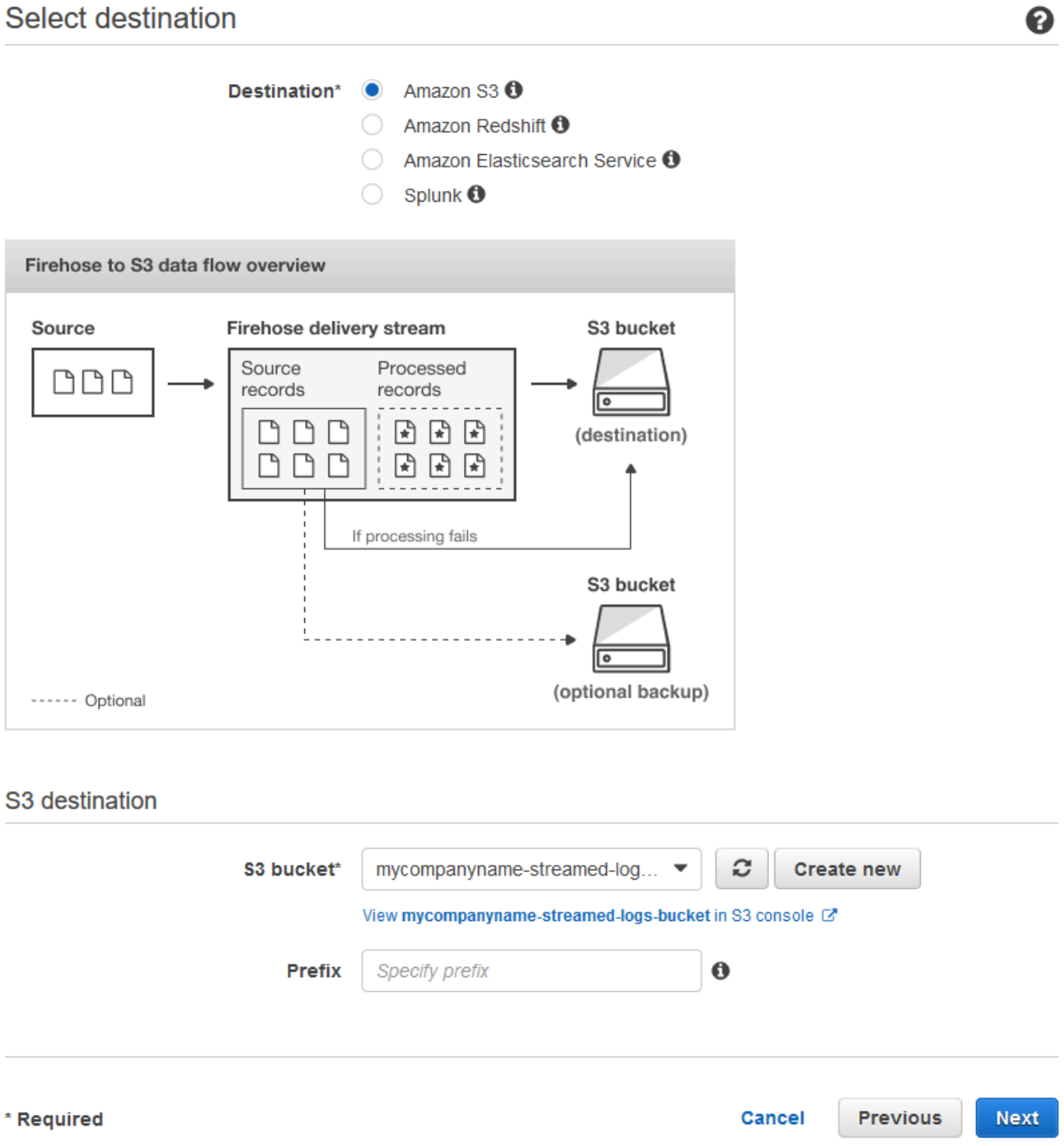
创建将流式记录存储在 Amazon S3 中的 Kinesis Data Firehose 传输流。

创建 Kinesis Data Firehose 传输流

1. 打开 Kinesis Data Firehose 控制台，网址为https://console.aws.amazon.com/firehose/。
2. 选择 Create Delivery Stream。
3. 在 Delivery stream name (传输流名称) 框中，输入 **log-delivery-stream**。
4. 对于 Source (源)，请选择 Direct PUT 或其他源。



5. 选择 Next。
6. 再次选择 Next (下一步)。
7. 对于目标，选择Amazon S3。
8. 对于 S3 bucket (S3 存储桶)，选择您在创建 Amazon S3 存储桶中创建的存储桶的名称。



9. 选择 Next。
10. 在 Buffer interval (缓冲间隔) 框中，输入 **60**。
11. 在 IAM role (IAM 角色) 下，选择 Create new or choose (新建或选择)。
12. 对于 IAM role (IAM 角色)，选择 `firehose-s3-access-role`。

13. 选择 Allow。

Configure settings

Configure buffer, compression, logging, and IAM role settings for your delivery stream.

S3 buffer conditions

Firehose buffers incoming records before delivering them to your S3 bucket. Record delivery will be triggered once either of these conditions has been satisfied. Learn more

Buffer size* 5 MB

Specify a buffer size between 1-128 MB

Buffer interval* 60 seconds

Specify a buffer interval between 60-900 seconds

S3 compression and encryption

Firehose can compress records before delivering them to your S3 bucket. Compressed records can also be encrypted in the S3 bucket using a KMS master key. Learn more

S3 compression* (•) Disabled ( ) GZIP ( ) Snappy ( ) Zip

S3 encryption* (•) Disabled ( ) Enabled

Error logging

Firehose can log record delivery errors to CloudWatch Logs. If enabled, a CloudWatch log group and corresponding log streams are created on your behalf. Learn more

Error logging* ( ) Disabled (•) Enabled

IAM role

Firehose uses an IAM role to access your specified resources, such as the S3 bucket and KMS key. Learn more

IAM role* firehose-s3-access-role

Create new or choose

14. 选择 Next。
15. 选择 Create delivery stream (创建传输流)。

创建 Amazon EC2 实例以运行适用于 Windows 的 Kinesis 代理

创建使用 Windows Kinesis Agent 通过 Kinesis Data Firehose 流式传输日志记录的 EC2 实例。

创建 EC2 实例

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 按照 Amazon EC2 Windows 实例入门中的说明操作，并使用以下额外步骤：

   - 对于实例的 IAM role (IAM 角色)，选择 `kinesis-agent-instance-role`。
   - 如果您还没有连接到互联网的公共 Virtual Private Cloud (VPC)，请按照使用 Amazon EC2 进行设置中的适用于 Windows 实例的 Amazon EC2 用户指南。
   - 创建或使用一个安全组，该安全组仅允许从您的计算机或组织的计算机访问实例。有关更多信息，请参阅 。使用 Amazon EC2 进行设置中的适用于 Windows 实例的 Amazon EC2 用户指南。
   - 如果您指定现有密钥对，请确保有权访问密钥对的私有密钥。或者，创建新的密钥对，并将私有密钥保存在安全位置。
   - 在继续之前，请等待实例运行并完成两项运行状况检查。
   - 您的实例需要一个公有 IP 地址。如果尚未分配一个，请按照弹性 IP 地址中的适用于 Windows 实例的 Amazon EC2 用户指南。

后续步骤

步骤 2: 安装、配置和运行适用于 Windows 的运行代理

步骤 2: 安装、配置和运行适用于 Windows 的运行代理

在此步骤中，您使用 AWS 管理控制台远程连接到您在创建 Amazon EC2 实例以运行适用于 Windows 的 Kinesis 代理。然后，您在实例上安装 Microsoft Windows 的 Amazon Kinesis Agent，为 Windows 创建和部署 Kinesis Agent 的配置文件，并启动冬奥会服务。

1. 通过 Remore Destorage Destorage Destorage Destorage Data (RDP) 中的说明操作来远程连接到实例。步骤 2: 连接到您的实例中的适用于 Windows 实例的 Amazon EC2 用户指南。

2. 在实例上，使用 Windows Server Manager 为用户和管理员禁用 Microsoft Internet Explorer 增强的安全配置。有关更多信息，请参阅 Microsoft TechNet 网站上的如何关闭 Internet Explorer 增强的安全配置。
3. 在实例上，安装和配置适用于 Windows 的 Kinesis 代理。有关更多信息，请参阅 安装适用于 Windows 的 Kinesis 代理。
4. 在实例上，使用记事本为 Windows 配置文件创建 Kinesis 代理。将此文件保存到 %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json。将以下内容添加到此配置文件中：

```
{
  "Sources": [
    {
      "Id": "JsonLogSource",
      "SourceType": "DirectorySource",
      "RecordParser": "SingleLineJson",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "InitialPosition": 0
    }
  ],
  "Sinks": [
    {
      "Id": "FirehoseLogStream",
      "SinkType": "KinesisFirehose",
      "StreamName": "log-delivery-stream",
      "Region": "us-east-1",
      "Format": "json",
      "ObjectDecoration": "ComputerName={ComputerName};DT={timestamp:yyyy-MM-dd
 HH:mm:ss}"
    }
  ],
  "Pipes": [
    {
      "Id": "JsonLogSourceToFirehoseLogStream",
      "SourceRef": "JsonLogSource",
      "SinkRef": "FirehoseLogStream"
    }
  ]
}
```

此文件将配置为 Windows Kinesis Agent 以发送 JSON 格式的日志记录从c:\logsource\目录（source）传输流中的 Kinesis Data Firehose 传输流（名为log-delivery-stream（sink)。在将每条日志记录流式传输到 Kinesis Data Firehose 之前，会使用两个额外的键/值对（包含计算机名称和时间戳）对其进行增强。

5. 创建 c:\LogSource\ 目录，并使用 Notepad 在该目录中创建具有以下内容的 test.log 文件：

```
{ "Message": "Copasetic message 1", "Severity": "Information" }
{ "Message": "Copasetic message 2", "Severity": "Information" }
{ "Message": "Problem message 2", "Severity": "Error" }
{ "Message": "Copasetic message 3", "Severity": "Information" }
```

6. 在提升的 PowerShell 会话中，使用以下命令启动 AWSKinesisTap 服务：

```
Start-Service -ServiceName AWSKinesisTap
```

7. 使用文件资源管理器，导航到 %PROGRAMDATA%\Amazon\AWSKinesisTap\logs 目录。打开最新的日志文件。日志文件应类似于以下内容：

```
2018-09-28 23:51:02.2472 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.AWS.AWSEventSinkFactory.
2018-09-28 23:51:02.2784 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Windows.PerformanceCounterSinkFactory.
2018-09-28 23:51:02.5753 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Core.DirectorySourceFactory.
2018-09-28 23:51:02.5909 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.ExchangeSource.ExchangeSourceFactory.
2018-09-28 23:51:02.5909 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Uls.UlsSourceFactory.
2018-09-28 23:51:02.5909 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Windows.WindowsSourceFactory.
2018-09-28 23:51:02.9347 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Core.Pipes.PipeFactory.
2018-09-28 23:51:03.5128 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.AutoUpdate.AutoUpdateFactory.
2018-09-28 23:51:03.5440 Amazon.KinesisTap.Hosting.LogManager INFO Performance
 counter sink  started.
2018-09-28 23:51:03.7628 Amazon.KinesisTap.Hosting.LogManager INFO
 KinesisFirehoseSink id FirehoseLogStream for StreamName log-delivery-stream
 started.
```

```
2018-09-28 23:51:03.7784 Amazon.KinesisTap.Hosting.LogManager INFO Connected source
 JsonLogSource to sink FirehoseLogStream
2018-09-28 23:51:03.7940 Amazon.KinesisTap.Hosting.LogManager INFO DirectorySource
 id JsonLogSource watching directory C:\LogSource\ with filter *.log started.
```

此日志文件指示服务已启动，并且现在正在从 `c:\LogSource\` 目录收集日志记录。每一行都被解析为一个 JSON 对象。计算机名称和时间戳的键/值对将添加到每个对象中。然后将其流式传输到 Kinesis Data Firehose。

8. 在一两分钟后，Amazon S3 航到您在创建 Amazon S3 存储桶使用 AWS 管理控制台。确保在控制台上选择了正确的区域。

   在该存储桶中，有一个针对当前年份的文件夹。打开该文件夹以显示当前月份的文件夹。打开该文件夹以显示当天的文件夹。打开该文件夹以显示当前小时（用 UTC 表示）的文件夹。打开该文件夹以显示一个或多个以名称 `log-delivery-stream` 开头的项目。

9. 打开最新项目的内容以确认日志记录已成功存储在 Amazon S3 中并进行了所需的增强。如果一切配置正确，则内容看起来类似于以下内容：

```
{"Message":"Copasetic message 1","Severity":"Information","ComputerName":"EC2AMAZ-
ABCDEFGH","DT":"2018-09-28 23:51:04"}
{"Message":"Copasetic message 2","Severity":"Information","ComputerName":"EC2AMAZ-
ABCDEFGH","DT":"2018-09-28 23:51:04"}
{"Message":"Problem message 2","Severity":"Error","ComputerName":"EC2AMAZ-
ABCDEFGH","DT":"2018-09-28 23:51:04"}
{"Message":"Copasetic message 3","Severity":"Information","ComputerName":"EC2AMAZ-
ABCDEFGH","DT":"2018-09-28 23:51:04"}
```

10. 有关解决以下任何问题的信息，请参阅针对微软 Windows 的 Amazon Kinesis 代理疑难解答：

    - Windows 日志文件中包含错误。

- Amazon S3 中的预期文件夹或项目不存在。
- Amazon S3 项目的内容不正确。

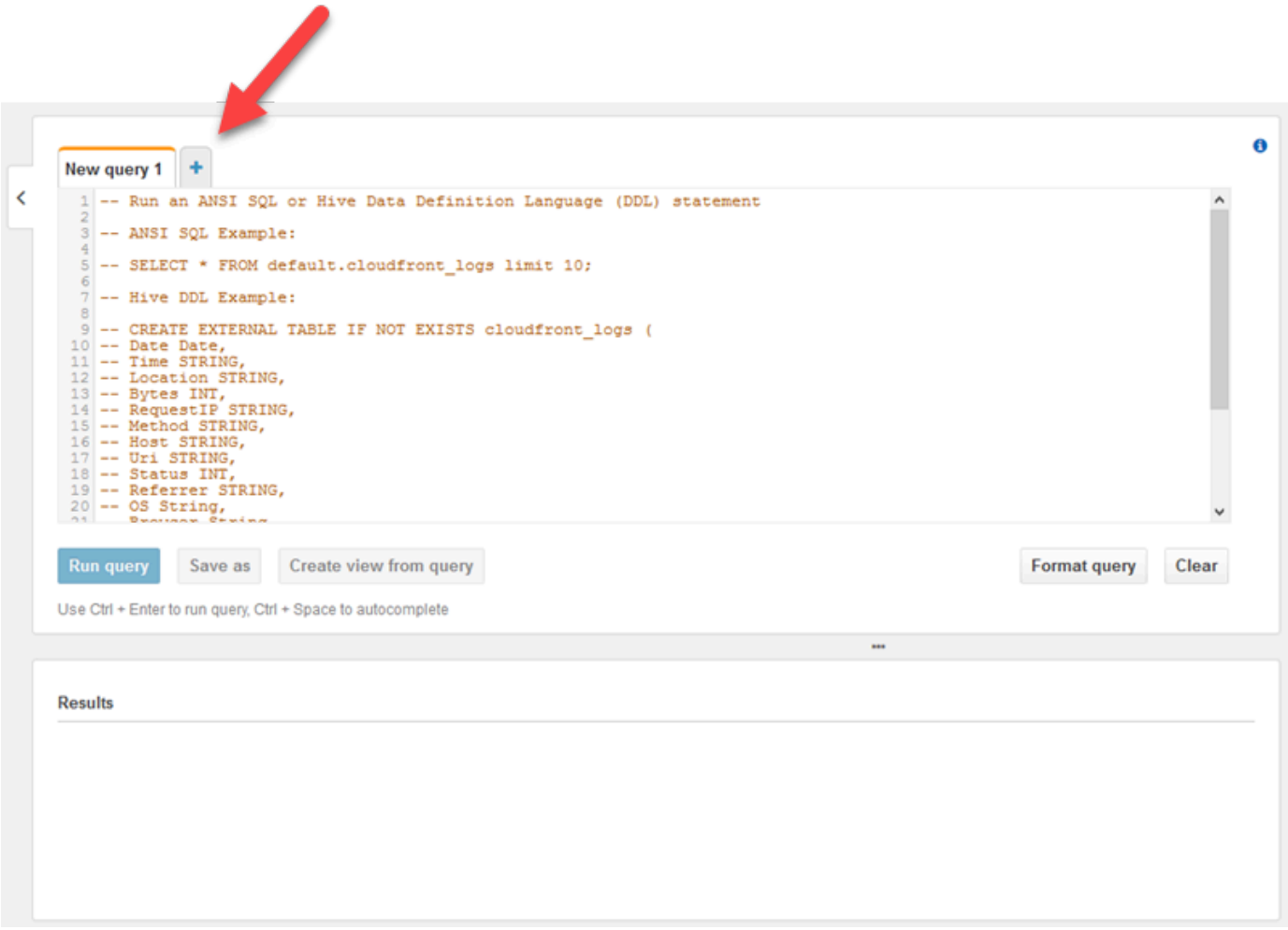
后续步骤

步骤 3: 在 Amazon S3 中查询日志数据

步骤 3: 在 Amazon S3 中查询日志数据

在此 Amazon Kinesis 代理微软 Windows 的最后一步教程中，您使用 Amazon Athena 查询 Amazon Simple Storage Service (Amazon S3) 中存储的日志数据。

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 选择加号 (+) 创建新的查询窗口。

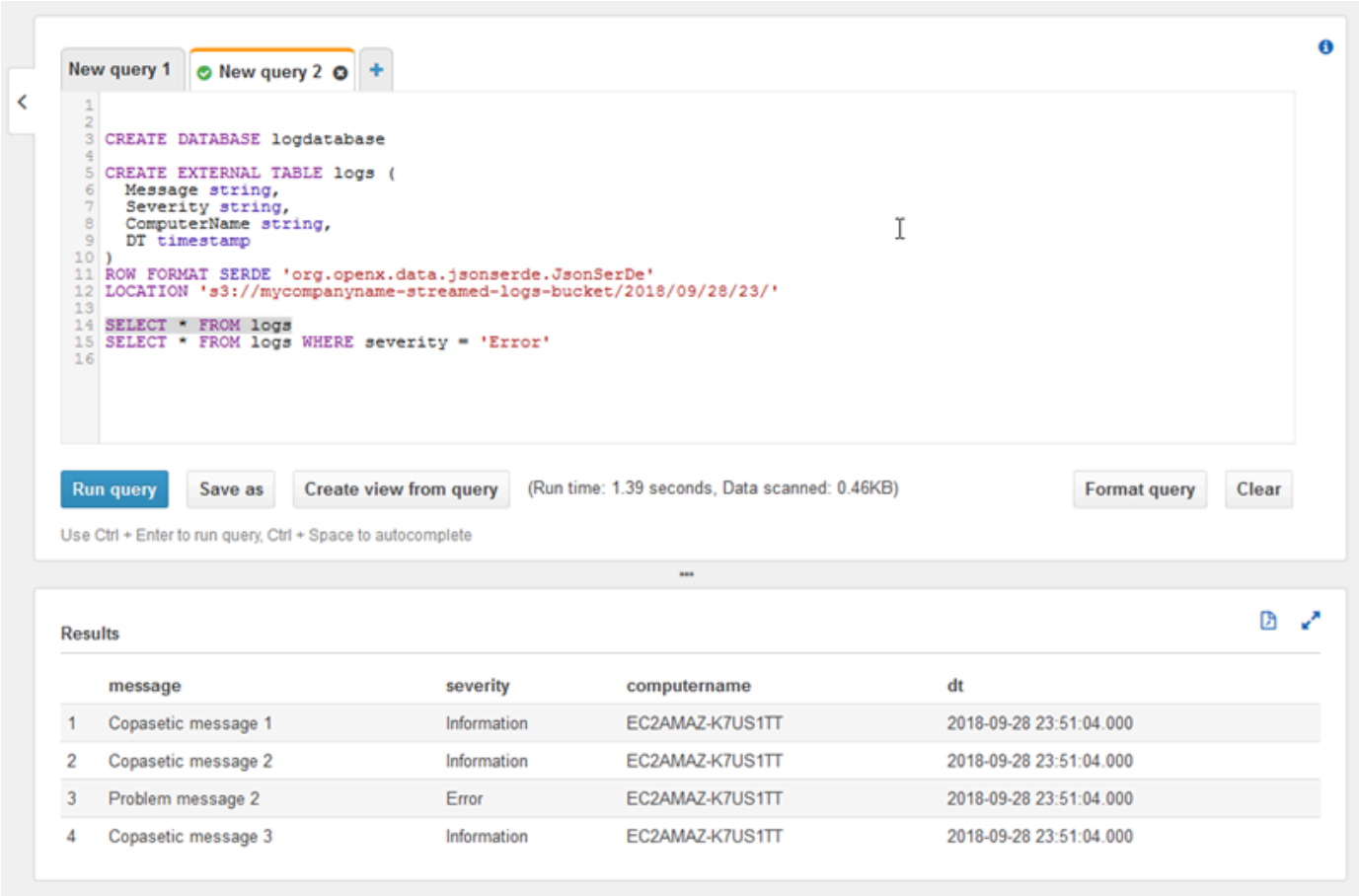


3. 在查询窗口中输入以下文本：

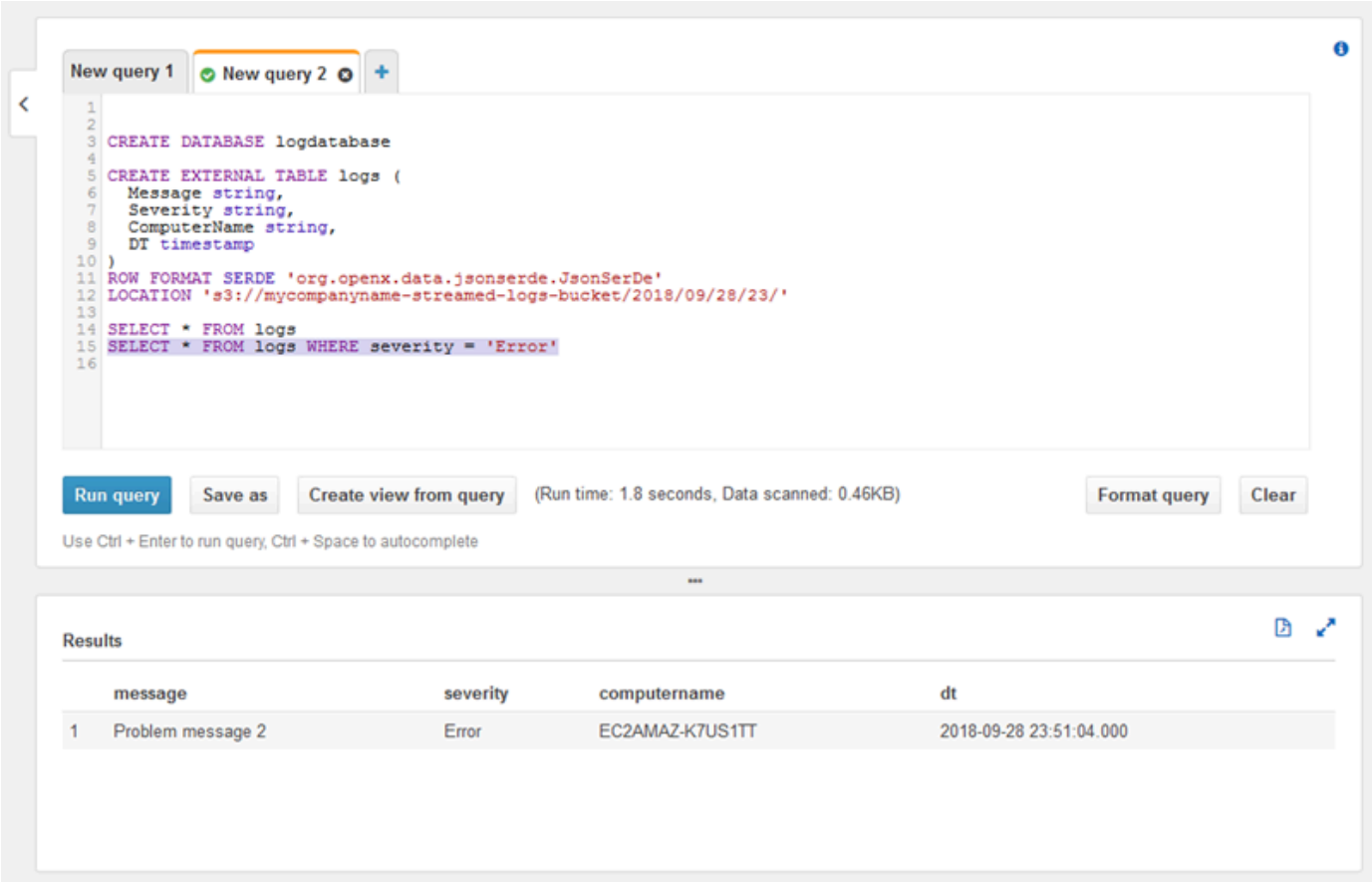
```
CREATE DATABASE logdatabase

CREATE EXTERNAL TABLE logs (
  Message string,
  Severity string,
  ComputerName string,
  DT timestamp
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://bucket/year/month/day/hour/'

SELECT * FROM logs
SELECT * FROM logs WHERE severity = 'Error'
```

将 *bucket* 替换为您在创建 Amazon S3 存储桶中创建的存储桶的名称。Replace*year*、*month*、*day*和*hour*分别为 Amazon S3 日志文件的创建时间（用 UTC 表示）中的年、月、日和小时。

4. 选择 CREATE DATABASE 语句的文本，然后选择 Run query (运行查询)。这将在 Athena 中创建日志数据库。
5. 选择 CREATE EXTERNAL TABLE 语句的文本，然后选择 Run query (运行查询)。这将创建一个 Athena 表，该表使用日志数据引用 S3 存储桶，并将 JSON 的架构映射到 Athena 表的架构。
6. 选择第一个 SELECT 语句的文本，然后选择 Run query (运行查询)。这将显示表中的所有行。



7. 选择第二个 SELECT 语句的文本，然后选择 Run query (运行查询)。这仅显示表中表示具有 `Error` 级别严重性的日志记录的行。此类查询从可能很大的日志记录集中查找有趣的日志记录。



后续步骤

使用 AWS 管理控制台清除本教程中创建的资源：

1. 终止 EC2 实例（参阅 Amazon EC2 Windows 实例入门中的步骤 3）。

   > **Important**
   >
   > 如果您启动的实例不在AWS 免费套餐，则您需要为实例支付相关费用，直到您终止该实例。

2. 删除 Kinesis Data Firehose 传输流。

   a. 打开 Kinesis Data Firehose 控制台，网址为https://console.aws.amazon.com/firehose/。

   b. 选择您创建的传输流。

   c. 选择 Delete。

   d. 选择 Delete delivery stream (删除传输流)。

3. 删除 S3 存储桶。有关说明，请参阅如何删除 S3 存储桶？中的Amazon Simple Storage Service 控制台用户指南。

有关更多信息，请参阅以下主题：

- 为微软 Windows 配置 Amazon Kinesis 运行代理
- 什么是 Amazon Kinesis Data Firehose？
- 什么是 Amazon S3？
- 什么是 Amazon Athena？

针对微软 Windows 的 Amazon Kinesis 代理疑难解答

请按照以下说明来诊断并纠正的使用适用于 Microsoft Windows 的 Amazon Kinesis 代理程序时的问题。

主题

没有数据从桌面或服务器流式传输到预期的 AWS 服务

Symptoms

当您检查由配置为从适用于 Windows 的 Kinesis 代理接收数据流的各种 AWS 服务所托管的日志、事件和指标时，不会流式传输任何数据。

Causes

有多种可能的原因会导致此问题：

- 源、接收器或管道配置不正确。
- 适用于 Windows 的 Kinesis 代理的身份验证配置不正确。
- Windows Kinesis 代理的授权配置不正确。
- `DirectorySource` 声明中提供的正则表达式存在错误。
- 为 `DirectorySource` 声明指定了一个不存在的目录。
- 正在向 AWS 服务提供的值无效，然后会拒绝来自适用于 Windows 的 Kinesis 代理的请求。
- 接收器引用指定的或隐式 AWS 区域中不存在的资源。
- 为 `WindowsEventLogSource` 声明指定了无效的查询。

- 为源的 InitialPosition 键/值对指定了无效值。
- appsettings.json 配置文件不符合该文件的 JSON 架构。
- 数据将流式传输到与 AWS 管理控制台中选择的区域不同的区域。
- 适用于 Windows 的 Kinesis 代理未正确安装或未运行。

Resolutions

要解决数据不进行流式处理的问题，请执行以下步骤：

1. 检查 Kinesis 代理程序中的 Windows 日志%PROGRAMDATA%\Amazon\AWSKinesisTap\logs目录。搜索字符串 ERROR。
   a. 如果未加载源或接收器，请执行以下操作：
      i. 检查错误消息，并找到源或接收器的 Id。
      ii. 检查与 %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 配置文件中的 Id 对应的源或接收器声明，以查找与找到的错误消息相关的任何错误。有关更多信息，请参阅为微软 Windows 配置 Amazon Kinesis 运行代理。
      iii. 更正与错误相关的任何配置文件问题。
      iv. 停止和启动 AWSKinesisTap 实例。然后，查看最新的日志文件以验证配置问题是否已解决。
   b. 如果错误消息指示未找到管道的 SourceRef 或 SinkRef，请执行以下操作：
      i. 记下管道 Id。
      ii. 检查 %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 配置文件中与记下的 Id 对应的管道声明。确保对于您要引用的源和接收器声明，SourceRef 和 SinkRef 键/值对的值正确拼写为 Id。更正任何拼写错误或拼写错误。如果配置文件中缺少源或接收器声明，请添加声明。有关更多信息，请参阅 为微软 Windows 配置 Amazon Kinesis 运行代理。
      iii. 停止和启动 AWSKinesisTap 实例。然后，查看最新的日志文件以验证配置问题是否已解决。
   c. 如果错误消息指示某个特定的 IAM 用户或角色未被授权执行某些操作，请执行以下操作：
      i. 确保适用于 Windows 的 Kinesis 代理正在使用的是正确的 IAM 用户或角色。如果不是，请查看接收器安全配置，并调整适用于 Windows 的 Kinesis 代理身份验证的方式，以确保使用的是正确的 IAM 用户或角色。
      ii. 如果使用的是正确的 IAM 用户或角色，请使用 AWS 管理控制台检查与该用户或角色关联的策略。对于适用于 Windows 的 Kinesis 代理访问的所有 AWS 资源，确保用户或角色具有相应错误消息中提到的所有权限。有关更多信息，请参阅 配置授权。

iii. 停止和启动 AWSKinesisTap 实例。然后，查看最新的日志文件以验证安全问题是否已解决。

d. 如果错误消息指示在解析 %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 配置文件中包含的正则表达式时存在参数错误，请执行以下操作：

i. 查看配置文件中的正则表达式。

ii. 验证正则表达式的语法。有几个网站可用于验证正则表达式，您也可以使用以下命令行检查 DirectorySource 源声明的正则表达式：

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
ktdiag.exe /r sourceId
```

将 *sourceId* 替换为具有错误正则表达式的 DirectorySource 源声明的 Id 键/值对的值。

iii. 对配置文件中的正则表达式进行任何必要的更正，以使其有效。

iv. 停止和启动 AWSKinesisTap 实例。然后，查看最新的日志文件以验证配置问题是否已解决。

e. 如果错误消息指示在解析 %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 配置文件中不包含的正则表达式时存在参数错误，并且该正则表达式与特定接收器有关，请执行以下操作：

i. 在配置文件中找到接收器声明。

ii. 验证与 AWS 服务明确相关的键/值对是否在使用符合该服务的验证规则的名称。例如，CloudWatch Logs 组名必须仅包含使用正则表达式指定的特定字符集。[\.\-_/#A-Za-z0-9]+。

iii. 更正接收器声明的键/值对中的任何无效名称，并确保在 AWS 中正确配置了这些资源。

iv. 停止和启动 AWSKinesisTap 实例。然后，查看最新的日志文件以验证配置问题是否已解决。

f. 如果错误消息指示由于参数为空或缺少参数而无法加载源或接收器，请执行以下操作：

i. 注意源或接收器的 Id。

ii. 找到与 %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 配置文件中已记下的 Id 匹配的源或接收器声明。

iii. 查看源或接收器声明中提供的键/值对，与相关接收器类型的 为微软 Windows 配置 Amazon Kinesis 运行代理 文档中的源或接收器类型要求进行比较。将缺少的任何必需键/值对添加到源或接收器声明中。

iv. 停止和启动 AWSKinesisTap 实例。然后，查看最新的日志文件以验证配置问题是否已解决。

g. 如果错误消息指示目录名称无效，请执行以下操作：

i. 在 %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 配置文件中找到无效的目录名称。

ii. 确保此目录是否存在并包含应流式传输的日志文件。

iii. 更正配置文件中指定的目录名称中的任何拼写错误。

iv. 停止和启动 AWSKinesisTap 实例。然后，查看最新的日志文件以验证配置问题是否已解决。

h. 如果错误消息指示某个资源不存在：

i. 找到 %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 配置文件中接收器声明中不存在的资源的资源引用。

ii. 使用 AWS 管理控制台在接收器声明中应使用的正确 AWS 区域中找到资源。将其与配置文件中指定的内容进行比较。

iii. 更改配置文件中的接收器声明，使其具有正确的资源名称和正确的区域。

iv. 停止和启动 AWSKinesisTap 实例。然后，查看最新的日志文件以验证配置问题是否已解决。

i. 如果错误消息指示特定 WindowsEventLogSource 的查询无效，请执行以下操作：

i. 在 %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 配置文件中，找到与错误消息中相应内容具有相同 Id 的 WindowsEventLogSource 声明。

ii. 验证源声明中 Query 键/值对的值是否符合 事件查询和事件 XML。

iii. 对查询进行任何更改以使其符合规范。

iv. 停止和启动 AWSKinesisTap 实例。然后，查看最新的日志文件以验证配置问题是否已解决。

j. 如果错误消息指示存在无效的初始位置，请执行以下操作：

i. 在 %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 配置文件中，找到与错误消息中相应内容具有相同 Id 的源声明。

ii. 更改源声明中 InitialPosition 键/值对的值以符合允许值，如书签配置中所述。

iii. 停止和启动 AWSKinesisTap 实例。然后，查看最新的日志文件以验证配置问题是否已解决。

2. 确保 %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 配置文件符合 JSON 架构。

a. 在命令提示符窗口中，调用以下行：

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
%PROGRAMFILES%\Amazon\AWSKinesisTap\ktdiag.exe /c
```

b. 更正使用 %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 配置文件检测到的任何问题。

c. 停止和启动 AWSKinesisTap 实例。然后，查看最新的日志文件以验证配置问题是否已解决。

3. 更改日志记录级别以尝试获取更详细的日志记录信息。

a. 将 %PROGRAMFILES%\Amazon\AWSKinesisTap\nlog.xml 配置文件替换为以下内容：

```
<?xml version="1.0" encoding="utf-8" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.nlog-project.org/schemas/NLog.xsd NLog.xsd"
      autoReload="true"
      throwExceptions="false"
      internalLogLevel="Off" internalLogFile="c:\temp\nlog-internal.log" >

  <!--
  See https://github.com/nlog/nlog/wiki/Configuration-file
  for information on customizing logging rules and outputs.
   -->
  <targets>
    <!--
    add your targets here
    See https://github.com/nlog/NLog/wiki/Targets for possible targets.
    See https://github.com/nlog/NLog/wiki/Layout-Renderers for the possible layout
 renderers.
    -->

    <target name="logfile"
            xsi:type="File"
            layout="${longdate} ${logger} ${uppercase:${level}} ${message}"
            fileName="${specialfolder:folder=CommonApplicationData}/Amazon/
KinesisTap/logs/KinesisTap.log"
     maxArchiveFiles="90"
     archiveFileName="${specialfolder:folder=CommonApplicationData}/Amazon/
KinesisTap/logs/Archive-{################}.log"
     archiveNumbering="Date"
     archiveDateFormat="yyyy-MM-dd"
     archiveEvery="Day"
     />
  </targets>

  <rules>
    <logger name="*" minlevel="Debug" writeTo="logfile" />
  </rules>
</nlog>
```

b. 停止和启动 AWSKinesisTap 实例。然后检查最新的日志文件，以查看日志中是否有其他消息可以帮助诊断和解决问题。

4. 确保您正在查看 AWS 管理控制台中正确区域中的资源。

5. 确保 Windows Kinesis 代理已安装并且正在运行。
   a. 在 Windows 中，选择开始，然后导航到控制面板、管理工具、服务。
   b. 查找 AWSKinesisTap 服务。
   c. 如果看不到 AWSKinesis 服务，请按照中的说明安装适用于 Windows 的 Kinesis 代理程序。适用于 Microsoft Windows 的 Amazon Kinesis 代理入门。
   d. 如果能看到该服务，请确定该服务是否正在运行。如果未运行，请打开该服务的上下文（右键单击）菜单，然后选择开始。
   e. 通过检查 `%PROGRAMDATA%\Amazon\AWSKinesisTap\logs` 目录中的最新日志文件来验证该服务是否已启动。

适用于

该信息适用于适用于 Windows 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.

预期的数据有时会丢失

Symptoms

适用于 Windows 的 Kinesis 代理大多数时候会成功地传输数据，但偶尔会丢失一些数据。

Causes

有多种可能的原因会导致此问题：

- 未使用书签功能。
- 根据服务的当前配置，超出了 AWS 服务的数据速率限制。
- AWS 服务的 API 调用速率限制将根据当前的`appsettings.json`配置文件和 AWS 账户限制。

Resolutions

要解决丢失数据的问题，请执行以下步骤：

1. 考虑使用 书签配置 中记录的书签功能。它有助于确保最终发送所有数据，即使 Windows Kinesis 代理停止并启动也是如此。
2. 使用适用于 Windows 的内置指标的 Kinesis 代理来发现问题：

a. 启用适用于 Windows 的 Kinesis 代理指标的流式传输，如中所述为 Windows 度量管道配置 Kinesis 代理。
b. 如果一个或多个接收器存在大量不可恢复的错误，请确定每秒发送的字节数或记录数。然后，确定相应数值是否位于为要在其中流式传输数据的区域和账户中的那些 AWS 服务配置的限制内。
c. 超出限制时，要么降低所发送数据的速率或数量，请求提高限制，要么增加分片（如果适用）。
d. 在进行调整后，继续监视的 Windows Kinesis 代理的内置指标，以确保这种情况已得到解决。

有关 Kinesis Data Streams 限制的更多信息，请参阅Kinesis Data Streams 限制中的Kinesis Data Streams 开发人员指南。有关 Kinesis Data Firehose 限制的更多信息，请参阅Amazon Kinesis Data Firehose 限制。

适用于

该信息适用于适用于 Windows 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.

数据到达时采用了错误格式

Symptoms

数据采用错误格式到达 AWS 服务。

Causes

有多种可能的原因会导致此问题：

- `appsettings.json` 配置文件中接收器声明的 `Format` 键/值对的值不正确。
- `DirectorySource` 声明中 `RecordParser` 键/值对的值不正确。
- 使用 `Regex` 记录解析程序的 `DirectorySource` 声明中的正则表达式不正确。

Resolutions

要解决格式不正确的问题，请执行以下步骤：

1. 查看 `%PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json` 配置文件中的接收器声明。

2. 确保为每个接收器声明指定了正确的 `Format` 键/值对值。有关更多信息，请参阅 接收器声明。
3. 如果具有 `DirectorySource` 声明的源通过管道连接到为 `Format` 键/值对指定 `xml` 或 `json` 值的接收器，请确保这些源指定 `RecordParser` 键/值对的以下值之一：
   - `SingleLineJson`
   - `Regex`
   - `SysLog`
   - `Delimited`

   其他记录解析程序仅基于文本，并且无法与需要 XML 或 JSON 格式的接收器一起正常工作。
4. 如果 `DirectorySource` 源类型未正确解析日志记录，请在命令提示符窗口中调用以下行以验证 `DirectorySource` 声明中指定的时间戳和正则表达式键/值对：

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
ktdiag.exe /r sourceID
```

   将 *`sourceID`* 替换为似乎没有正常工作的 `DirectorySource` 源声明的 `Id` 键/值对的值。更正 `ktdiag.exe` 报告的任何问题。

适用于

该信息适用于适用于 Windows 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.

性能问题

Symptoms

安装并启动适用于 Windows 的 Kinesis 代理后，应用程序和服务的延迟会增加。

Causes

有多种可能的原因会导致此问题：

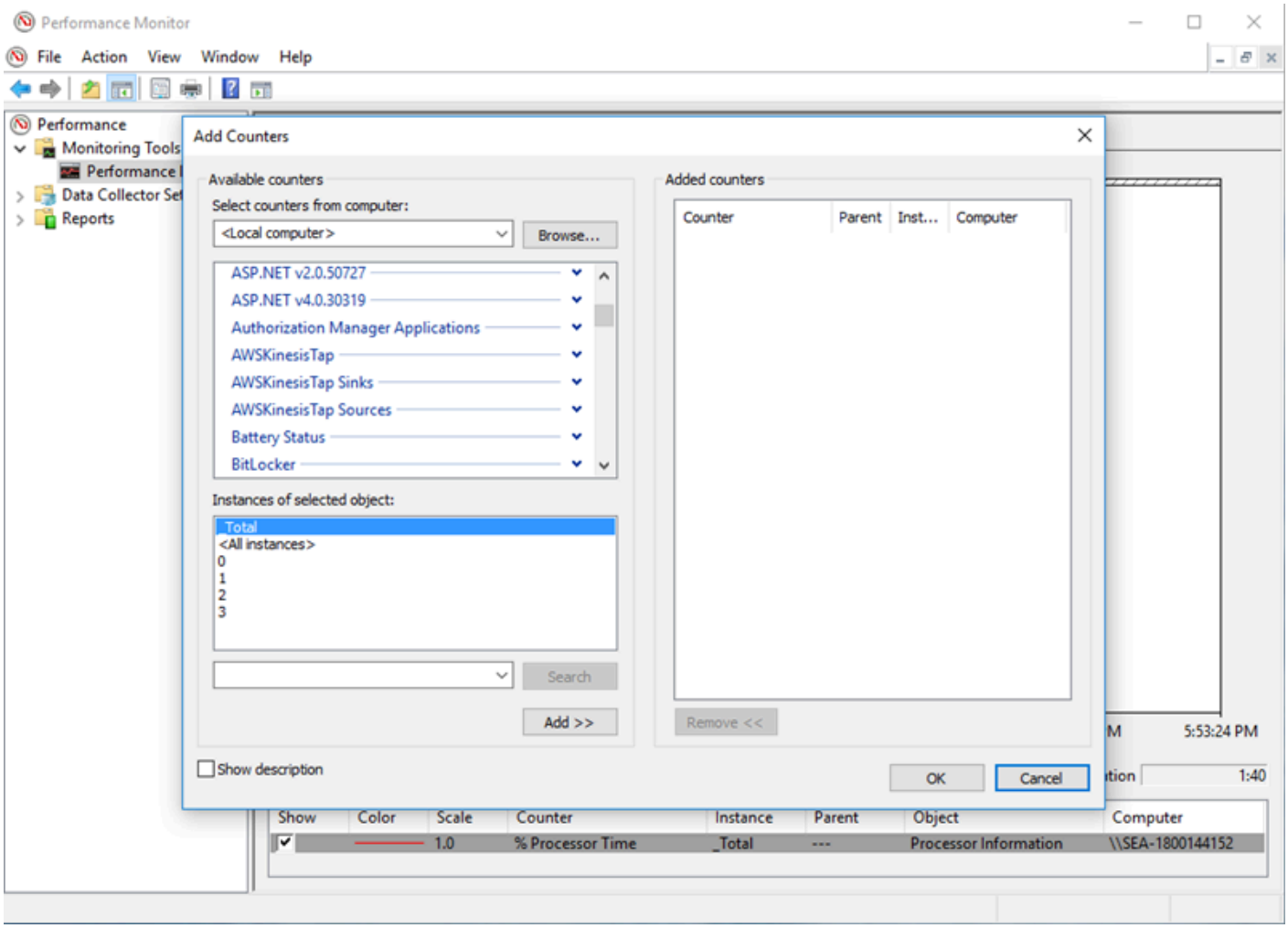
- 运行 Windows Kinesis 代理的计算机没有足够的容量来流式传输所需的数据量。
- 不必要的数据正在流式传输到一个或多个 AWS 服务。
- 适用于 Windows 的 Kinesis 代理正在将数据流式传输到 AWS 服务，但这些服务没有配置这么高的数据速率。

- 适用于 Windows 的 Kinesis 代理正在调用 AWS 服务的操作，该账户的 API 调用速率限制太低。

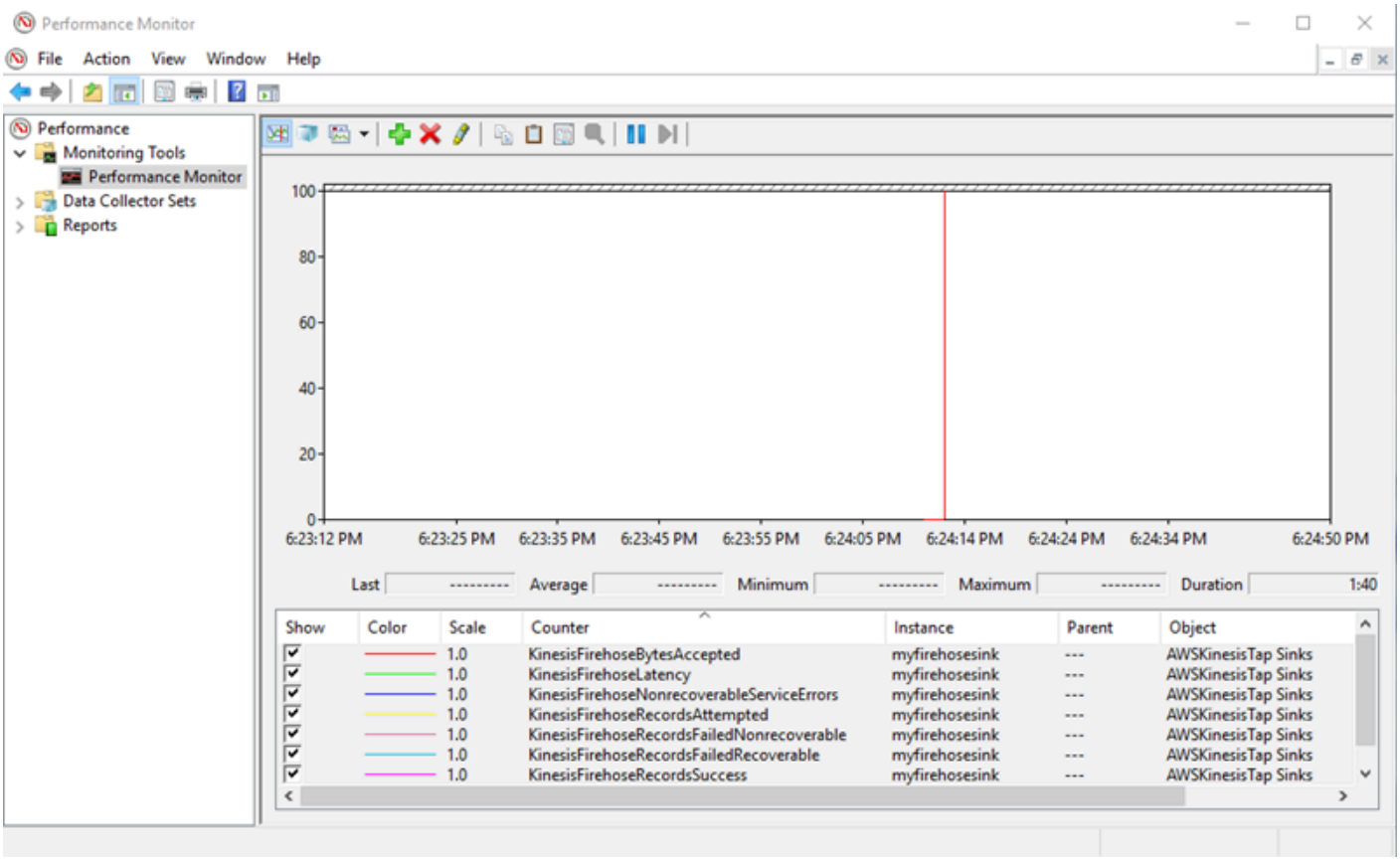
Resolutions

要解决性能问题，请执行以下步骤：

1. 使用 Windows 资源监视器应用程序检查内存、CPU、磁盘和网络使用情况。如果需要使用 Kinesis Agent 的 Windows 流式传输大量数据，则可能需要在一些区域中配置具有更高容量的计算机，具体取决于配置。
2. 您可以使用筛选功能减少记录的数据量：
   - 请参阅 WindowsEventLogSource 配置 中的 `Query` 键/值对。
   - 请参阅 配置管道中的管道筛选。
   - 请参阅 Amazon CloudWatch 指标筛选，请参阅CloudWatch 接收器配置)。
3. 使用 Windows 性能监视器应用程序查看 Windows 指标的 Kinesis 代理程序或者将这些指标流式传输到 CloudWatch 视器 (参阅Windows 内置指标源的 Kinesis 代理)。在 Windows 性能监视器应用程序中，您可以为 Windows 接收器和源添加 Kinesis 代理计数器。它们列在 AWSKinesisTap Sinks (AWSKinesisTap 接收器) 和 AWSKinesisTap Sources (AWSKinesisTap 源) 计数器类别下。



例如，要诊断 Kinesis Data Firehose 的性能问题，需要将Kinesis Firehose 接收器性能计数器。



如果存在大量可恢复的错误，请检查中的 Windows 日志的最新 Kinesis 代理程序。%PROGRAMDATA%\Amazon\AWSKInesisTap\logs目录。如果 `KinesisStream` 或 `KinesisFirehose` 接收器出现限制，请执行以下操作：

- 如果由于流式传输数据速度过快而出现限制，请考虑提高 Kinesis 数据流的分片数。有关更多信息，请参阅 。重新分片、扩展和并行处理中的Kinesis Data Streams 开发人员指南。
- 考虑提高 Kinesis Data Streams 的 API 调用限制，或者在 API 调用受到限制时增加接收器的缓冲区大小。有关更多信息，请参阅 。Kinesis Data Streams 限制中的Kinesis Data Streams 开发人员指南。
- 如果数据流式传输速度过快，请考虑请求提高 Kinesis Data Firehose 传输流的速率限制。或者，如果 API 调用正受到限制，则请求增加 API 调用限制（请参阅 Amazon Kinesis Data Firehose 限制）或者增加接收器的缓冲区大小。
- 增加动能数据流的分片数量或者增加 Kinesis 数 Kinesis Data Firehose 传输流的速率限制后，修改 Windows 的 Kinesis 代理`appsettings.json`配置文件来增加接收器的每秒记录数或每秒字节数。否则，适用于 Windows 的 Kinesis 代理无法利用增加的限制。

适用于

该信息适用于适用于 Windows 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.

磁盘空间不足

Symptoms

适用于 Windows 的 Kinesis 代理正在运行的计算机上一个或多个磁盘驱动器上的磁盘空间非常少。

Causes

有多种可能的原因会导致此问题：

- 适用于 Windows 日志记录配置文件不正确。
- 适用于 Windows 的 Kinesis 代理持久队列配置不正确。
- 其他某个应用程序或服务正在占用磁盘空间。

Resolutions

要解决磁盘空间问题，请执行以下步骤：

- 如果包含 Windows 日志文件的 Kinesis 代理的磁盘上的磁盘空间不足，请检查日志文件目录（通常为`%PROGRAMDATA%\Amazon\AWSKinesisTap\logs`)。确保保留合理数量的日志文件，并确保日志文件的大小合理。您可以控制适用于 Windows 日志的 Kinesis 代理的位置、保留和详细程度，方法是编辑`%PROGRAMFILES%\Amazon\AWSKinesisTap\Nlog.xml`配置文件。
- 启用接收器排队功能后，请检查使用该功能的接收器声明。确保 `QueuePath` 键/值对引用具有足够空间的磁盘驱动器，以能够容纳使用 `QueueMaxBatches` 键/值对指定的最大数量的批次。如果无法做到这一点，则减小 `QueueMaxBatches` 键/值对的值，以便数据轻松适合指定的磁盘驱动器的剩余磁盘空间。
- 使用 Windows 文件资源管理器找到占用磁盘空间的文件，并传输或删除多余的文件。更改占用大量磁盘空间的应用程序或服务的配置。

适用于

该信息适用于适用于 Windows 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.

故障排除工具

除了验证配置文件之外，您可以使用ktdiag.exe工具，该工具在配置和使用适用于 Windows 的 Kinesis 代理时提供了其他几种诊断和解决问题的功能。ktdiag.exe 工具位于 %PROGRAMFILES%\Amazon\AWSKinesisTap 目录中。

- 如果您认为具有特定文件模式的日志文件正在写入目录但未被处理，请使用/w开关以验证是否已检测到这些更改。例如，假设您希望带有 *.log 文件名模式的日志文件被写入 c:\foo 目录。执行 ktdiag.exe 工具时，可以使用 /w 开关并指定目录和文件模式：

  ```
  cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
  ktdiag /w c:\foo *.log
  ```

  如果正在写入日志文件，您可以看到类似于以下内容的输出：

  ```
  Type any key to exit this program...
  File: c:\foo\log1.log ChangeType: Created
  File: c:\foo\log1.log ChangeType: Deleted
  File: c:\foo\log1.log ChangeType: Created
  File: c:\foo\log1.log ChangeType: Changed
  File: c:\foo\log1.log ChangeType: Changed
  File: c:\foo\log1.log ChangeType: Changed
  File: c:\foo\log1.log ChangeType: Changed
  ```

  如果没有发生此类输出，则表示在写入日志时存在应用程序或服务问题，或者存在安全配置问题而不是的问题的 Windows Kinesis 代理问题。如果正在出现这样的输出，但适用于 Windows 的 Kinesis 代理仍然没有明显处理日志，请参阅没有数据从桌面或服务器流式传输到预期的 AWS 服务。

- 有时只会偶尔写入日志，但验证 Windows 的 Kinesis 代理是否正常运行会很有用。使用 /log4net 开关模拟使用 Log4net 库写入日志的应用程序；例如：

  ```
  cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
  KTDiag.exe /log4net c:\foo\log2.log
  ```

  这会将 Log4net 样式的日志文件写入 c:\foo\log2.log 日志文件，并在按下某个键之前不断添加新的日志条目。您可以在文件名后选择性地指定其他开关来配置多个选项：

  锁定：-lm、-li 或 -le

  您可以指定以下锁定开关之一来控制日志文件的锁定方式：

`-lm`

日志文件使用最少量的锁定，从而可以最大程度地访问日志文件。

`-li`

只有同一进程中的线程才能同时访问日志。

`-le`

一次只有一个线程可以访问日志。这是默认模式。

`-tn:##`

指定日志条目写入操作之间的*##*数。默认值为 1000 毫秒（1 秒）。

`-sm:##`

指定每个日志条目的*##*数。默认值为 1000 字节。

`-bk:number`

指定一次写入日志条目的*##*。默认值为 1。

- 有时，模拟写入 Windows 事件日志的应用程序很有用。使用 `/e` 开关在 Windows 事件日志中写入日志条目；例如：

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
KTDiag.exe /e Application
```

这会将日志条目写入 Windows 应用程序事件日志，直到按下一个键。您可以选择在日志名称后面指定以下附加选项：

`-tn:##`

指定日志条目写入操作之间的*##*数。默认值为 1000 毫秒（1 秒）。

`-sm:##`

指定每个日志条目的*##*数。默认值为 1000 字节。

`-bk:number`

指定一次写入日志条目的*##*。默认值为 1。

为 Windows 插件创建 Kinesis 代理

对于大多数情况下，不需要为微软 Windows 插件创建 Amazon Kinesis 代理。Windows 的 Kinesis Agent 高度可配置，并包含功能强大的源和接收器，例如`DirectorySource`和`KinesisStream`，这对于大多数情况都足够了。有关现有源和接收器的详细信息，请参阅为微软 Windows 配置 Amazon Kinesis 运行代理。

对于不常见的场景，可能需要使用定制插件扩展 Windows 的 Kinesis 代理。这样的一些场景包括：

- 使用 `Regex` 或 `Delimited` 记录解析程序打包复杂的 `DirectorySource` 声明，使其易于在多种不同类型的配置文件中应用。
- 创建并非基于文件或者超过了现有记录解析程序所提供解析功能的新源。
- 为 AWS 服务创建当前不支持的接收器。

主题

适用于 Windows 插件的 Kinesis 代理入门

关于自定义插件没有特殊要求。所有现有源和接收器使用的机制，与自定义插件在 Windows 启动时加载的机制相同，在读取`appsettings.json`配置文件。

当 Windows 的 Kinesis 代理程序集时，操作按照以下顺序进行：

1. Windows Kinesis 代理扫描安装目录中的程序集（`%PROGRAMFILES%\Amazon\AWSKinesisTap`），用于实现`IFactory<T>`接口中定义`Amazon.KinesisTap.Core`程序集。此接口定义在`Amazon.KinesisTap.Core\Infrastructure\IFactory.cs`在 Kinesis 代理程序的 Windows 源代码中。
2. Windows 的 Kinesis 代理加载包含这些类的程序集并调用`RegisterFactory`方法。
3. 适用于 Windows 的 Kinesis 代理加载`appsettings.json`配置文件。对于配置文件中的每个源和接收器，检查 `SourceType` 和 `SinkType` 键/值对。如果有工厂的注册名称与 `SourceType` 和 `SinkType` 键/值对中的值相同，则调用这些工厂上的 `CreateInstance` 方法。`CreateInstance`

方法将配置和其他信息作为 IPluginContext 对象传递。CreateInstance 方法负责配置和初始化插件。

要使插件正确工作，必须有创建插件的已注册工厂类，并且插件类本身必须已定义。

Windows Kinesis 代理位于https://github.com/awslabs/kinesis-agent-windows。

为 Windows 插件工厂实施 Kinesis 代理

按照以下步骤来实施 Windows 插件工厂的 Kinesis 代理。

为 Windows 插件工厂创建一个 Kinesis 代理

1. 创建定位到 .NET Framework 4.6 的 C# 库项目。
2. 添加对 Amazon.KinesisTap.Core 程序集的引用。此程序集位于%PROGRAMFILES%\Amazon\AWSKinesisTap目录之后的 Kinesis 代理进行 Windows 安装。
3. 使用 NuGet 安装 Microsoft.Extensions.Configuration.Abstractions 程序包。
4. 使用 NuGet 安装 System.Reactive 程序包。
5. 使用 NuGet 安装 Microsoft.Extensions.Logging 程序包。
6. 创建为源实施 IFactory<IEventSource> 或为接收器实施 IFactory<IEventSink> 的工厂类。添加 RegisterFactory 和 CreateInstance 方法。

   例如，以下代码创建 Windows 插件工厂的 Kinesis 代理，该工厂创建生成随机数据的源：

```
using System;
using Amazon.KinesisTap.Core;
using Microsoft.Extensions.Configuration;

namespace MyCompany.MySources
{
    public class RandomSourceFactory : IFactory<ISource>
    {
        public void RegisterFactory(IFactoryCatalog<ISource> catalog)
        {
            catalog.RegisterFactory("randomsource", this);
        }

        public ISource CreateInstance(string entry, IPlugInContext context)
        {
```

```
            IConfiguration config = context.Configuration;

            switch (entry.ToLower())
            {
                case "randomsource":
                    string rateString = config["Rate"];
                    string maxString = config["Max"];
                    TimeSpan rate;
                    int max;

                    if (string.IsNullOrWhiteSpace(rateString))
                    {
                        rate = TimeSpan.FromSeconds(30);
                    }
                    else
                    {
                        if (!TimeSpan.TryParse(rateString, out rate))
                        {
                            throw new Exception($"Rate {rateString} is invalid for
RandomSource.");
                        }
                    }

                    if (string.IsNullOrWhiteSpace(maxString))
                    {
                        max = 1000;
                    }
                    else
                    {
                        if (!int.TryParse(maxString, out max))
                        {
                            throw new Exception($"Max {maxString} is invalid for
RandomSource.");
                        }
                    }

                    return new RandomSource(rate, max, context);
                default:
                    throw new ArgumentException($"Source {entry} is not
recognized.", entry);
            }
        }
    }
```

```
}
```

当您最终希望增强工厂以创建不同类型的实例时，switch 语句在 CreateInstance 方法中使用。

要创建接收器工厂（该工厂创建不执行任何操作的接收器），请使用类似于下文的类：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Amazon.KinesisTap.Core;
using Microsoft.Extensions.Configuration;

namespace MyCompany.MySinks
{
    public class NullSinkFactory : IFactory<IEventSink>
    {
        public void RegisterFactory(IFactoryCatalog<IEventSink> catalog)
        {
            catalog.RegisterFactory("nullsink", this);
        }

        public IEventSink CreateInstance(string entry, IPlugInContext context)
        {
            IConfiguration config = context.Configuration;

            switch (entry.ToLower())
            {
                case "nullsink":
                    return new NullSink(context);
                default:
                    throw new Exception("Unrecognized sink type {entry}.");
            }
        }
    }
}
```

实现 Windows 插件 Kinesis 代理程序

按照以下步骤来实施 Windows 插件源的 Kinesis 代理。

创建 Windows 插件 Kinesis 代理程序

1. 将实施 `IEventSource<out T>` 接口的类添加到以前为源创建的项目。

   例如，使用以下代码来定义生成随机数据的源：

```
using System;
using System.Reactive.Subjects;
using System.Timers;
using Amazon.KinesisTap.Core;
using Microsoft.Extensions.Logging;

namespace MyCompany.MySources
{
    public class RandomSource : EventSource<RandomData>, IDisposable
    {
        private TimeSpan _rate;
        private int _max;
        private Timer _timer = null;
        private Random _random = new Random();
        private ISubject<IEnvelope<RandomData>> _recordSubject = new
 Subject<IEnvelope<RandomData>>();



        public RandomSource(TimeSpan rate, int max, IPlugInContext context) :
 base(context)
        {
            _rate = rate;
            _max = max;
        }

        public override void Start()
        {
            try
            {
                CleanupTimer();
                _timer = new Timer(_rate.TotalMilliseconds);
                _timer.Elapsed += (Object source, ElapsedEventArgs args) =>
```

```
                {
                    var data = new RandomData()
                    {
                        RandomValue = _random.Next(_max)
                    };
                    _recordSubject.OnNext(new Envelope<RandomData>(data));
                };
                _timer.AutoReset = true;
                _timer.Enabled = true;
                _logger?.LogInformation($"Random source id {this.Id} started with
rate {_rate.TotalMilliseconds}.");
            }
            catch (Exception e)
            {
                _logger?.LogError($"Exception during start of RandomSource id
{this.Id}: {e}");
            }
        }

        public override void Stop()
        {
            try
            {
                CleanupTimer();
                _logger?.LogInformation($"Random source id {this.Id} stopped.");

            }
            catch (Exception e)
            {
                _logger?.LogError($"Exception during stop of RandomSource id
{this.Id}: {e}");
            }
        }

        private void CleanupTimer()
        {
            if (_timer != null)
            {
                _timer.Enabled = false;
                _timer?.Dispose();
                _timer = null;
            }
        }
```

```
        public override IDisposable Subscribe(IObserver<IEnvelope<RandomData>>
 observer)
        {
            return this._recordSubject.Subscribe(observer);
        }

        public void Dispose()
        {
            CleanupTimer();
        }
    }
}
```

在本示例中，RandomSource 类继承自 EventSource<T> 类，因为它提供了 Id 属性。虽然本示例不支持书签，此基类也可用于实施该功能。信封提供了一种存储元数据和包装任意数据用于流式传输到接收器的方法。RandomData 类在下一步中定义，表示来自此源的输出对象的类型。

2. 添加类到以前定义的项目，该项目中包含从源流式传输的数据。

   例如，随机数据的容器可以定义如下所示：

```
namespace MyCompany.MySources
{
    public class RandomData
    {
        public int RandomValue { get; set; }
    }
}
```

3. 编译以前定义的项目。
4. 将程序集复制到 Windows 的 Kinesis 代理的安装目录。
5. 创建或更新appsettings.json配置文件，并将其放在 Windows 的 Kinesis 代理的安装目录中。
6. 停止然后重新启动 Windows 的 Kinesis 代理。
7. 检查当前 Kinesis 代理的 Windows 日志文件（通常位于%PROGRAMDATA%\Amazon \AWSKinesisTap\logs目录）以确保自定义源插件没有问题。
8. 确保数据到达目标 AWS 服务。

有关如何扩展DirectorySource功能来实施解析特定日志格式的功能，请参阅Amazon.KinesisTap.Uls\UlsSourceFactory.cs和Amazon.KinesisTap.Uls \UlsLogParser.cs在 Kinesis 代理程序的 Windows 源代码中。

有关如何创建提供书签功能的源的示例，请参阅Amazon.KinesisTap.Windows\WindowsSourceFactory.cs和Amazon.KinesisTap.Windows\EventLogSource.cs在 Kinesis 代理程序的 Windows 源代码中。

实现 Windows 插件接收器的 Kinesis 代理

按照以下步骤来实施 Windows 插件接收器的 Kinesis 代理。

为 Windows 插件接收器创建 Kinesis 代理程序

1. 将类添加到之前定义的项目（该项目实施 IEventSink 接口）。

   例如，以下代码实施接收器，该接收器除了对记录到达进行日志记录之外不执行任何操作，随后丢弃到达的记录。

```
using Amazon.KinesisTap.Core;
using Microsoft.Extensions.Logging;

namespace MyCompany.MySinks
{
    public class NullSink : EventSink
    {
        public NullSink(IPlugInContext context) : base(context)
        {
        }

        public override void OnNext(IEnvelope envelope)
        {
            _logger.LogInformation($"Null sink {Id} received
 {GetRecord(envelope)}.");
        }

        public override void Start()
        {
            _logger.LogInformation($"Null sink {Id} starting.");
        }

        public override void Stop()
        {
            _logger.LogInformation($"Null sink {Id} stopped.");
        }
    }
```

```
}
```

在此示例中，NullSink 接收器类继承自 EventSink 类，因为它提供将记录转换为不同序列化格式（例如 JSON 和 XML）的功能。

2. 编译以前定义的项目。
3. 将程序集复制到 Windows 的 Kinesis 代理的安装目录。
4. 创建或更新appsettings.json配置文件，并将其放在 Windows 的 Kinesis 代理的安装目录中。例如，要使用 RandomSource 和 NullSink 自定义插件，您可以使用以下 appsettings.json 配置文件：

```
{
  "Sources": [
  {
 "Id": "MyRandomSource",
 "SourceType": "RandomSource",
 "Rate": "00:00:10",
 "Max": 50
  }

  ],
  "Sinks": [
  {
    "Id": "MyNullSink",
 "SinkType": "NullSink",
 "Format": "json"
  }
  ],
  "Pipes": [
    {
   "Id": "MyRandomToNullPipe",
   "SourceRef": "MyRandomSource",
   "SinkRef": "MyNullSink"
 }
  ]
}
```

此配置创建发送 RandomData 的实例的源，其中 RandomValue 每隔 10 秒设置为 0 到 50 之间的随机数字。它创建一个接收器，将传入 RandomData 实例转换为 JSON、记录该 JSON 然后丢弃实例。请确保包含之前所定义项目中的示例工厂、RandomSource 源类以及 NullSink 接收器类以使用示例配置文件。

5. 停止然后重新启动 Windows 的 Kinesis 代理。
6. 检查当前 Kinesis 代理的 Windows 日志文件（通常位于%PROGRAMDATA%\Amazon\AWSKinesisTap\logs目录）以确保自定义接收器插件没有问题。
7. 确保数据到达目标 AWS 服务。由于示例 NullSink 不流式传输到 AWS 服务，您可以通过查找指示已收到记录的日志消息来验证接收器的操作正确。

   举例来说，您可以看到类似于下文的日志文件：

```
2018-10-18 12:36:36.3647 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.AWS.AWSEventSinkFactory.
2018-10-18 12:36:36.4018 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Windows.PerformanceCounterSinkFactory.
2018-10-18 12:36:36.4018 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory MyCompany.MySinks.NullSinkFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Core.DirectorySourceFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.ExchangeSource.ExchangeSourceFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Uls.UlsSourceFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Windows.WindowsSourceFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory MyCompany.MySources.RandomSourceFactory.
2018-10-18 12:36:36.9601 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Core.Pipes.PipeFactory.
2018-10-18 12:36:37.4694 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.AutoUpdate.AutoUpdateFactory.
2018-10-18 12:36:37.4807 Amazon.KinesisTap.Hosting.LogManager INFO Performance
 counter sink  started.
2018-10-18 12:36:37.6250 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink starting.
2018-10-18 12:36:37.6250 Amazon.KinesisTap.Hosting.LogManager INFO Connected source
 MyRandomSource to sink MyNullSink
2018-10-18 12:36:37.6333 Amazon.KinesisTap.Hosting.LogManager INFO Random source id
 MyRandomSource started with rate 10000.
2018-10-18 12:36:47.8084 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":14}.
2018-10-18 12:36:57.6339 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":5}.
2018-10-18 12:37:07.6490 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":9}.
```

```
2018-10-18 12:37:17.6494 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":47}.
2018-10-18 12:37:27.6520 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":25}.
2018-10-18 12:37:37.6676 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":21}.
2018-10-18 12:37:47.6688 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":29}.
2018-10-18 12:37:57.6700 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":22}.
2018-10-18 12:38:07.6838 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":32}.
2018-10-18 12:38:17.6848 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":12}.
2018-10-18 12:38:27.6866 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":46}.
2018-10-18 12:38:37.6880 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":48}.
2018-10-18 12:38:47.6893 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":39}.
2018-10-18 12:38:57.6906 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":18}.
2018-10-18 12:39:07.6995 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":6}.
2018-10-18 12:39:17.7004 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":0}.
2018-10-18 12:39:27.7021 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":3}.
2018-10-18 12:39:37.7023 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":19}.
```

如果您创建访问 AWS 服务的接收器，则可能会发现基类非常有用。对于使用`AWSBufferedEventSink`基类，请参阅`Amazon.KinesisTap.AWS\CloudWatchLogsSink.cs`在适用于 Windows 的 Kinesis 代理的源代码中。

适用于微软 Windows 的 Amazon Kinesis Agent 文档历史记录

API 版本：2018-10-15

下表介绍了对适用于微软 Windows 的 Amazon Kinesis 代理程序用户指南(本文件).

更新-历史记录-更改	更新-历史记录-描述	更新-历史记录-日期
主要文档更新	添加了有关 MSI 安装的说明。更新了目录源配置并添加了窗口七个日志轮询源。对于接收器配置，添加了本地文件系统同步配置；配置文件刷新 AwsdentialProject；有关文本装饰、解析汇属性中的变量、为汇配置 STS 区域终端节点、配置 VPC 终端节点以及配置备用代理服务器的信息。对于管道，添加了配置属性。	2021 年 2 月 23 日
更新了文档	更新了主题，指示 Amazon S3 位置规范区分大小写。	2018 年 11 月 7 日
初始版本 1.0.0.0.0.0.0.0.0.0.0.115	适用于 Windows 的 Kinesis 代理用户指南的第一个版本。	2018 年 11 月 5 日

AWS 词汇表

For the latest AWS terminology, see the AWS glossary in the AWS General Reference.

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。