Amazon Kinesis Data Analytics

Amazon Kinesis Data Analytics 开发者指南

Amazon Kinesis Data Analytics: Amazon Kinesis Data Analytics 开发者指南

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

Table of Contents

什么是 Amazon Kinesis Data Analytics for Apache Flink？

借助适用于 Apache Flink 的 Amazon Kinesis Data Analytics，您可以使用 Java、Scala 或 SQL 处理和分析流数据。该服务用于根据流式源创建和运行代码，以便执行时间序列分析，为实时控制面板提供信息以及创建实时指标。

您可以使用基于以下基于的开源库在 Kinesis Data Analytics 中构建 Java 和 Scala 应用程序。Apache Flink. Apache Flink 是处理数据流的常用框架和引擎。

> Note
>
> 尽管 Kinesis Data Analytics 支持使用 Scala 版本 2.12 编写的 Apache Flink 应用程序，但本指南只包含用 Java 编写的代码示例。

Kinesis Data Analytics 为 Apache Flink 应用程序提供基础设施。它实施一些核心功能，例如，预置计算资源、并行计算、自动扩展和应用程序备份（实施为检查点和快照）。您可以使用高级 Flink 编程功能（如操作符、函数、源和接收器），使用的方式与自行托管 Flink 基础设施时一样。

入门

您可以从创建持续读取和处理流数据的 Kinesis Data Analytics 应用程序开始。然后，使用所选的 IDE 编写代码，并使用实时流数据对其进行测试。您还可以配置 Kinesis Data Analytics 要将结果发送到的目标。

首先，我们建议您阅读以下章节：

- Apache Flink 的 Kinesis Data Analytics of Apache Flink：工作方式 (p. 2)
- 使用 Amazon Kinesis Data Analytics Apache Flink 入门 (DataStreamAPI) (p. 76)

Apache Flink 的 Kinesis Data Analytics of Apache Flink：工作方式

Amazon Kinesis Data Analytics of Apache Flink 是一项完全托管的 Amazon 服务，可让您使用 Apache Flink 应用程序处理流数据。

编程你的 Apache Flink 应用程序

Apache Flink 应用程序是使用 Apache Flink 框架创建的 Java 或 Scala 应用程序。您在本地创作并构建 Apache Flink 应用程序。

应用程序主要使用DataStreamAPI或者表 API. 您也可以使用其他 Apache Flink API，但在构建流应用程序时很少使用这些 API。

这两个 API 的功能如下：

DataStream API

Apache FlinkDataStreamAPI 编程模型基于两个组件：

- 数据流：数据记录连续流的结构化表示形式。
- 转换操作符：将一个或多个数据流作为输入，并生成一个或多个数据流以作为输出。

使用创建的应用程序DataStreamAPI 执行以下操作：

- 从数据源读取数据（例如 Kinesis 流或 Amazon MSK 主题）。
- 对数据应用转换，例如筛选、聚合或丰富。
- 将转换后的数据写入数据汇。

使用DataStreamAPI 可以用 Java 或 Scala 编写，也可以从 Kinesis 数据流、Amazon MSK 主题或自定义源中读取。

您的应用程序通过使用连接器. Apache Flink 使用以下类型的连接器：

- 源：用于读取外部数据的连接器。
- 接收器：用于写入外部位置的连接器。
- 运算符：用于处理应用程序内数据的连接器。

典型的应用程序包含至少一个具有源的数据流、一个具有一个或多个操作符的数据流以及至少一个数据接收器。

有关如何使用 DataStream API 的更多信息，请参阅DataStream API (p. 8)。

表 API

Apache Flink 表 API 编程模型基于以下组件：

- 表环境：用于创建和托管一个或多个表的基础数据的接口。
- 表：提供对 SQL 表或视图的访问权限的对象。
- 表来源：用于从外部来源读取数据，例如亚马逊 MSK 主题。
- 表函数：用于转换数据的 SQL 查询或 API 调用。
- 表接收器：用于将数据写入外部位置，如 Amazon S3 存储桶。

使用表 API 创建的应用程序执行以下操作：

- 创建`TableEnvironment`通过连接到`Table Source`.
- 在中创建表`TableEnvironment`使用 SQL 查询或表 API 函数。
- 使用表 API 或 SQL 对表运行查询
- 使用表函数或 SQL 查询对查询结果应用转换。
- 将查询或函数结果写入`Table Sink`.

使用表 API 的应用程序可以用 Java 或 Scala 编写，并可以使用 API 调用或 SQL 查询来查询数据。

有关使用表 API 的更多信息，请参阅表 API (p. 13).

创建 Kinesis Data Analytics 应用程序

Kinesis Data Analytics 应用程序是Amazon由 Kinesis Data Analytics 服务托管的资源。您的 Kinesis Data Analytics 应用程序托管您的 Apache Flink 应用程序并为其提供以下设置：

- 运行时属性 (p. 18)：可以提供给应用程序的参数。您可以更改这些参数，而无需重新编译应用程序代码。
- 容错能力 (p. 21)：应用程序如何从中断中恢复并重新启动。
- 日志记录和监控 (p. 204)：应用程序如何将事件记录到CloudWatch日志。
- 扩缩 (p. 27)：应用程序如何配置计算资源。

您可以 Kinesis Data Analytics 制台或使用Amazon CLI. 要开始创建 Kinesis Data Analytics 应用程序，请参阅入门 (DataStreamAPI) (p. 76).

创建 Apache Flink 应用程序的 Kinesis Data Analytics 应用程序

本主题包含有关创建适用于 Apache Flink 的 Kinesis Data Analytics 应用程序的信息。

本主题包含下列部分：

构建 Kinesis Data Analytics 应用程序代码

本节介绍用于为 Kinesis Data Analytics 应用程序构建应用程序代码的组件。

我们建议您将支持的最新 Apache Flink 版本用于应用程序代码。Kinesis Data Analytics 支持的最新 Apache Flink 版本是1.13.2. 有关升级 Kinesis Data Analytics 应用程序的信息，升级应用程序 (p. 260).

您可以使用 Apache Maven 构建应用程序代码。Apache Maven 项目使用 `pom.xml` 文件以指定它使用的组件的版本。

> Note
>
> Kinesis Data Analytics 支持最多 512 MB 大小的 JAR 文件。如果使用的 JAR 文件超过该大小，应用程序将无法启动。
> 有关解决方法，请参阅JAR 超过 512 MB (p. 336)。

将以下组件版本用于 Kinesis Data Analytics 应用程序：

组件	版本
Java	11（推荐）
Scala	2.12
适用于 Flink 的 Kinesis Data Analytics 运行时 (aws-kinesisanalytics-runtime)	1.2.0
Kinesis Data Analytics Flink 连接器 (aws-kinesisanalytics-flink)	2.0.0
AmazonKinesis 连接器（闪烁连接器-运动）	1.13.2
Apache Beam（仅限 Beam 应用程序）	使用 2.33.0 版本 2.12.2

> Note
>
> 在新区域中，我们仅支持服务可用时及以后的最新 Flink 版本。

对于一个例子`pom.xml`使用 Apache Flink 版本 1.13.2 的 Kinesis Data Analytics 应用程序的文件，请参阅Kinesis Data Analytics 入门应用程序。

有关创建 Kinesis Data Analytics 应用程序的信息Apache Beam请参阅使用 Apache Beam (p. 113).

指定应用程序的 Apache Flink 版本

在使用 Kinesis Data Analytics 版本 1.1.0 及更高版本时，您可以在编译应用程序时指定应用程序使用的 Apache Flink 版本。您可以使用 `-Dflink.version` 参数提供 Apache Flink 版本，如下所示：

```
mvn package -Dflink.version=1.13.2
```

有关使用旧版本的 Apache Flink 构建应用程序的信息，请参阅早期版本 (p. 258)。

创建 Kinesis Data Analytics 应用程序

在构建应用程序代码后，您可以执行以下操作以创建 Kinesis Data Analytics 应用程序：

- 上传应用程序代码：将应用程序代码上传到 Amazon S3 存储桶。在创建应用程序时，您可以指定应用程序代码的 S3 存储桶名称和对象名称。有关说明如何上传应用程序代码的教程，请参阅入门 (DataStreamAPI) (p. 76)教程中的the section called “上传 Apache Flink 流式处理 Java 代码” (p. 82)。
- 创建 Kinesis Data Analytics 应用程序：使用以下方法之一以创建 Kinesis Data Analytics 应用程序：
  - 使用创建 Kinesis Data Analytics 应用程序Amazon控制台：您可以使用Amazon控制台。

在使用控制台创建应用程序时，应用程序的相关资源（例如 CloudWatch 为您创建了日志流、IAM 角色和 IAM 策略）。

在使用控制台创建应用程序时，您可以从 Kinesis Analytics - Create application (Kinesis Analytics - 创建应用程序) 页面上的下拉列表中进行选择，以指定应用程序使用的 Apache Flink 版本。

有关如何使用控制台创建应用程序的教程，请参阅入门 (DataStreamAPI) (p. 76)教程中的the section called “创建并运行应用程序（控制台）” (p. 83)。

- 使用创建 Kinesis Data Analytics 应用程序AmazonCLI：您可以使用AmazonCLI。

  在使用 CLI 创建应用程序时，您还必须创建应用程序的相关资源（例如 CloudWatch 手动记录流、IAM 角色和 IAM 策略）。

  在使用 CLI 创建应用程序时，您可以使用 `CreateApplication` 操作的 `RuntimeEnvironment` 参数指定应用程序使用的 Apache Flink 版本。

  有关如何使用 CLI 创建应用程序的教程，请参阅入门 (DataStreamAPI) (p. 76)教程中的the section called “使用 CLI 创建和运行应用程序” (p. 86)。

  Note

  您无法更改现有应用程序的 `RuntimeEnvironment`。如果您需要更改现有应用程序的 `RuntimeEnvironment`，则必须删除该应用程序并重新创建。

启动 Kinesis Data Analytics 应用程序

在构建应用程序代码、将其上传到 S3 并创建 Kinesis Data Analytics 应用程序后，您可以启动应用程序。启动 Kinesis Data Analytics 应用程序通常需要几分钟时间。

可以使用以下方法之一以启动应用程序：

- Kinesis Data Analytics 用Amazon控制台：你可以通过选择运行你的应用运行在应用程序页面上Amazon控制台。
- Kinesis Data Analytics 用AmazonAPI：你可以使用StartApplicationaction.

验证 Kinesis Data Analytics 应用程序

您可以通过以下方式验证应用程序是否正常工作：

- 使用 CloudWatch 日志：您可以使用 CloudWatch 日志和 CloudWatch 记录 Insights 以验证应用程序是否正常运行。有关使用的信息 CloudWatch 使用 Kinesis Data Analytics 应用程序记录，请参阅日志记录和监控 (p. 204).
- 使用 CloudWatch 指标：您可以使用 CloudWatch 监控应用程序活动的指标，或监控应用程序用于输入或输出的资源（例如 Kinesis Data Firehose 传输流或 Amazon S3 存储桶）中的活动。有关 的更多信息 CloudWatch 指标，请参阅使用指标在 Amazon CloudWatch 用户指南。
- 监控输出位置：如果应用程序将输出写入到某个位置（例如 Amazon S3 存储桶或数据库），您可以在该位置中监控写入的数据。

运行 Apache Flink 应用程序的 Kinesis Data Analytics

本主题包含有关运行 Apache Flink 应用程序的 Kinesis Data Analytics 应用程序的信息。

运行 Kinesis Data Analytics 应用程序时，Kinesis Data Analytics 服务会创建 Apache Flink 作业。Apache Flink 作业是您的 Kinesis Data Analytics 应用程序的执行生命周期。Job 的执行及其使用的资源由作业管理器管理。作 Job 管理器将应用程序的执行分为任务。每个任务都由任务管理器管理。监控应用程序的性能时，可以检查每个任务管理器或作业管理器 Job 为一个整体的性能。

有关 Apache Flink 作业的信息，请参阅作业和计划中的Apache Flink 文档.

任务申请和 Job 状态

你的申请和应用程序的作业都有当前的执行状态：

- 应用程序状态：您的应用程序的当前状态描述了其执行阶段。应用程序状态包括以下内容：
  - 稳定的应用程序状态：在进行状态更改之前，您的应用程序通常会保持这些状态：
    - 准备就绪：在运行之前，新应用程序或已停止的应用程序处于 READY 状态。
    - 运行：成功启动的应用程序处于 RUNING 状态。
  - 暂时应用程序状态：处于这些状态的应用程序通常处于过渡到另一种状态的过程中。如果应用程序在一段时间内保持暂时状态，则可以使用StopApplication使用操作`Force`设置参数为`true`. 这些状态包括以下内容：
    - `STARTING`:发生在StartApplicationaction. 应用程序正在从`READY`到`RUNNING`状态。
    - 停止：发生在StopApplicationaction. 应用程序正在从`RUNNING`到`READY`状态。
    - `DELETING`:发生在DeleteApplicationaction. 该应用程序正在删除中。
    - `UPDATING`:发生在UpdateApplicationaction. 应用程序正在更新，并将过渡回`RUNNING`要么`READY`状态。
    - `AUTOSCALING`:该应用程序具有`AutoScalingEnabled`的财产 ParallelismConfiguration设置为`true`，而且该服务正在增加应用程序的并行性。当应用程序处于此状态时，唯一可以使用的有效 API 操作是StopApplication使用操作`Force`设置参数为`true`. 有关自动扩展的信息，请参阅自动扩展 (p. 29).
    - `FORCE_STOPPING`:发生在StopApplication动作是用`Force`设置参数为`true`. 该应用程序正在强制停止中。应用程序从`STARTING`、`UPDATING`、`STOPPING`，或者`AUTOSCALING`状态为`READY`状态。
    - `ROLLING_BACK`:发生在RollbackApplication行动被称为。该应用程序正在回滚到以前的版本。应用程序从`UPDATING`要么`AUTOSCALING`状态为`RUNNING`状态。
    - `ROLLED_BACK`:成功回滚应用程序后，这将成为您从中回滚的版本的状态。有关回滚应用程序的信息，请参阅RollbackApplication.
    - `MAINTENANCE`:当 Kinesis Data Analytics 将修补程序应用于应用程序时发生。有关更多信息，请参阅 维护 (p. 243)。

  您可以使用控制台或使用DescribeApplicationaction.
- Job 状态：当你的应用程序在`RUNNING`状态，您的作业的状态描述了其当前执行阶段。一份工作从`CREATED`状态，然后继续转到`RUNNING`启动时的状态。如果出现错误情况，您的应用程序将进入以下状态：
  - 对于使用 Apache Flink 1.11 及更高版本的应用程序，您的应用程序将输入`RESTARTING`状态。
  - 对于使用 Apache Flink 1.8 及更早版本的应用程序，您的应用程序将输入`FAILING`状态。

  然后，应用程序将进入`RESTARTING`要么`FAILED`状态，取决于是否可以重新启动作业。

  你可以通过检查你的申请来查看工作的状态CloudWatch状态变化记录。

批处理工作负载

Kinesis Data Analytics 支持运行 Apache Flink 批处理工作负载。在批处理作业中，当 Apache Flink 作业到达完成状态，Kinesis Data Analytics 应用程序状态设置为准备好了. 有关 Flink 作业状态的更多信息，请参阅作业和计划.

应用程序资

本节介绍应用程序使用的系统资源。了解 Kinesis Data Analytics 如何配置和使用资源将有助于您设计、创建和维护高性能稳定的 Kinesis Data Analytics 应用程序。

Kinesis Data Analytics 应用程序

Kinesis Data AnalyticsAmazon为托管 Apache Flink 应用程序创建一个环境的服务。Kinesis Data Analytics 服务使用名为的单元提供资源Kinesis 处理单元 (KPU).

一个 KPU 代表以下系统资源：

- 一个 CPU 核心
- 4 GB 内存，其中 1 GB 是本机内存，三 GB 是堆内存
- 50 GB 的磁盘空间

KPU 以不同的执行单元运行应用程序任务和子任务. 您可以将子任务视为线程。

可供应用程序使用的 KPU 数量等于应用程序`Parallelism`设置，除以应用程序的`ParallelismPerKPU`设置。

有关应用程序并行度的更多信息，请参阅扩缩 (p. 27).

Apache Flink 应用程序资源

Apache Flink 环境使用名为的单元为应用程序分配资源任务槽. 当 Kinesis Data Analytics 为您的应用程序分配资源时，它会将一个或多个 Apache Flink 任务插槽分配给单个 KPU。分配给单个 KPU 的插槽数量等于你的应用程序`ParallelismPerKPU`设置。有关任务槽的更多信息，请参阅作业计划中的Apache Flink 文档.

操作符并行度

您可以设置操作符可以使用的最大子任务数。这个值被称为操作符并行度. 默认情况下，应用程序中每个运算符的并行度等于应用程序的并行度。这意味着，默认情况下，如果需要，应用程序中的每个操作员都可以使用应用程序中的所有可用子任务。

您可以使用以设置应用程序中操作符的并行度`setParallelism`方法。使用此方法，您可以控制每个操作员一次可以使用的子任务数量。

有关操作符链的更多信息，请参阅任务链接和资源组中的Apache Flink 文档.

串联操作符

通常，每个运算符都使用单独的子任务来执行，但是如果多个运算符始终按顺序执行，则运行时可以将它们全部分配给同一个任务。此过程称为串联操作符.

如果几个顺序运算符都在同一个数据上运行，则可以将它们链接到一个任务中。以下是实现此所需的一些标准：

- 运营商进行一对一的简单转发。
- 运算符都具有相同的运算符并行度。

当您的应用程序将操作员链接到单个子任务中时，它会节省系统资源，因为该服务不需要执行网络操作和为每个操作员分配子任务。要确定应用程序是否使用操作员链接，请查看 Kinesis Data Analytics 控制台中的作业图。应用程序中的每个顶点代表一个或多个操作符。该图显示了作为单个顶点链接的运算符。

DataStream API

你的 Apache Flink 应用程序使用Apache FlinkDataStreamAPI以转换数据流中的数据。

本节包含以下主题：

- 在 Kinesis Data Analytics 中使用连接器以移动 Apache FlinkDataStreamAPI (p. 8)：这些组件在应用程序与外部数据源和目标之间移动数据。
- 使用 Apache Flink 在 Kinesis Data Analytics Data Analytics 中使用操作符以使用 DataStreamAPI (p. 12)：这些组件对应用程序中的数据元素进行转换或分组。
- 使用DataStreamAPI (p. 13)：本主题介绍 Kinesis Data Analytics 如何在使用DataStreamAPI。

在 Kinesis Data Analytics 中使用连接器以移动 Apache FlinkDataStreamAPI

在 Amazon Kinesis Data Analytics in Apache FlinkDataStreamAPI，连接器是将数据移入和移出 Kinesis Data Analytics 应用程序的软件组件。连接器是灵活集成的组件，以使您能够读取文件和目录。连接器包含用于与 Amazon 服务和第三方系统交互的完整模块。

连接器类型包括：

- 源 (p. 8)：从 Kinesis 数据流、文件或其他数据源中向应用程序提供数据。
- 接收器 (p. 9)：将数据从应用程序发送到 Kinesis Data Streams、Kinesis Data Firehose 传输流或其他数据目标。
- 异步 I/O (p. 12)：提供对数据源（例如数据库）的异步访问以丰富流事件。

可用的连接器

Apache Flink 框架包含用于从各种源中访问数据的连接器。有关 Apache Flink 框架中可用的连接器的信息，请参阅连接器中的Apache Flink 文档.

将流数据源添加到 Apache Flink 的 Kinesis Data Analytics

Apache Flink 提供连接器以从文件、套接字、集合和自定义源中读取。在应用程序代码中，您可以使用 Apache Flink 源以从流中接收数据。本节介绍了可用于 Amazon 服务的源。

Kinesis Data Streams

这些区域有：`FlinkKinesisConsumer`源从 Amazon Kinesis 数据流中向应用程序提供流数据。

创建`FlinkKinesisConsumer`

以下代码示例说明了如何创建 `FlinkKinesisConsumer`：

```
Properties inputProperties = new Properties();
inputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
inputProperties.setProperty(ConsumerConfigConstants.STREAM_INITIAL_POSITION, "LATEST");

DataStream<string> input = env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new
 SimpleStringSchema(), inputProperties));
```

有关使用 `FlinkKinesisConsumer` 的更多信息，请参阅下载并检查 Apache Flink 流式处理 Java 代码 (p. 81)。

创建FlinkKinesisConsumer使用 EFO 消费者

这些区域有：FlinkKinesis现在使用者支持增强型扇出功能（EFO）.

如果 Kinesis 消费者使用 EFO，Kinesis Data Streams 服务将为其提供自己的专用带宽，而不是让消费者与其他消费者共享直播的固定带宽。

有关将 EFO 与 Kinesis 使用者一起使用的更多信息，请参阅翻转 128：的增强型扇出功能AmazonKinesis 使用者.

您可以通过在 Kinesis 使用器上设置以下参数来启用 EFO 使用者：

- 记录_PUBLISHER _ 类型：将该参数设置为EFO让您的应用程序使用 EFO 使用者访问 Kinesis 数据流数据。
- EFO_消费ER_NAME：将此参数设置为在此流的使用者中唯一的字符串值。在同一 Kinesis Data Stream 中重复使用消费者姓名将导致以前使用该名称的消费者被终止。

配置FlinkKinesisConsumer要使用 EFO，请将以下参数添加到使用者：

```
consumerConfig.putIfAbsent(RECORD_PUBLISHER_TYPE, "EFO");
consumerConfig.putIfAbsent(EFO_CONSUMER_NAME, "basic-efo-flink-app");
```

有关使用 EFO 消费者的 Kinesis Data Analytics 应用程序的示例，请参阅EFO cumer (p. 149).

Amazon MSK

这些区域有：FlinkKafkaConsumer源从 Amazon MSK 主题中向应用程序提供流数据。

创建FlinkKafkaConsumer

以下代码示例说明了如何创建 FlinkKafkaConsumer：

```
Properties inputProperties = new Properties();
inputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
inputProperties.setProperty("bootstrap.servers", "Cluster Bootstrap Broker String");
inputProperties.setProperty("security.protocol", "SSL");
inputProperties.setProperty("ssl.truststore.location", "/usr/lib/jvm/java-11-amazon-
corretto/lib/security/cacerts");
inputProperties.setProperty("ssl.truststore.password", "changeit");

DataStream<string> input = env.addSource(new FlinkKafkaConsumer<>("MyMSKTopic", new
 SimpleStringSchema(), inputProperties));
```

有关使用 FlinkKafkaConsumer 的更多信息，请参阅MSK 复制 (p. 145)。

Apache Flink 在 Kinesis Data Analytics 中使用接收器编写数据

在应用程序代码中，可以使用Apache Flink 接收器将 Apache Flink 流中的数据写入Amazon服务，例如 Kinesis Data Streams。

Apache Flink 提供了文件和套接字接收器以及自定义接收器。以下接收器可用于Amazon：

Kinesis Data Streams

Apache Flink 提供了有关Kinesis Data Streams 连接器在 Apache Flink 文档中。

有关使用 Kinesis 数据流进行输入和输出的应用程序的示例，请参阅入门 (DataStreamAPI) (p. 76).

Amazon S3

您可以使用 Apache Flink `StreamingFileSink` 以将对象写入到 Amazon S3 存储桶中。

有关如何将对象写入到 S3 的示例，请参阅the section called “S3 接收器” (p. 135)。

Kinesis Data Firehose

这些区域有：`FlinkKinesisFirehoseProducer`是一个可靠且可扩展的 Apache Flink 接收器，可以使用Kinesis Data Firehose服务。本节介绍了如何设置 Maven 项目以创建和使用 `FlinkKinesisFirehoseProducer`。

主题

创建`FlinkKinesisFirehoseProducer`

以下代码示例说明了如何创建 `FlinkKinesisFirehoseProducer`：

```
Properties outputProperties = new Properties();
outputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);

FlinkKinesisFirehoseProducer<String> sink = new
 FlinkKinesisFirehoseProducer<>(outputStreamName, new SimpleStringSchema(),
 outputProperties);
```

`FlinkKinesisFirehoseProducer` 代码示例

以下代码示例说明了如何创建和配置`FlinkKinesisFirehoseProducer`并将数据从 Apache Flink 数据流发送到 Kinesis Data Firehose link 服务。

```
package com.amazonaws.services.kinesisanalytics;

import
 com.amazonaws.services.kinesisanalytics.flink.connectors.config.ProducerConfigConstants;
import
 com.amazonaws.services.kinesisanalytics.flink.connectors.producer.FlinkKinesisFirehoseProducer;
import com.amazonaws.services.kinesisanalytics.runtime.KinesisAnalyticsRuntime;
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisProducer;

import org.apache.flink.streaming.connectors.kinesis.config.ConsumerConfigConstants;

import java.io.IOException;
import java.util.Map;
import java.util.Properties;



public class StreamingJob {

 private static final String region = "us-east-1";
 private static final String inputStreamName = "ExampleInputStream";
 private static final String outputStreamName = "ExampleOutputStream";
```

```
private static DataStream<String> createSourceFromStaticConfig(StreamExecutionEnvironment
env) {
 Properties inputProperties = new Properties();
 inputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
 inputProperties.setProperty(ConsumerConfigConstants.STREAM_INITIAL_POSITION, "LATEST");

 return env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new
SimpleStringSchema(), inputProperties));
}

private static DataStream<String>
createSourceFromApplicationProperties(StreamExecutionEnvironment env) throws IOException {
 Map<String, Properties> applicationProperties =
KinesisAnalyticsRuntime.getApplicationProperties();
 return env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new
SimpleStringSchema(),
   applicationProperties.get("ConsumerConfigProperties")));
}

private static FlinkKinesisFirehoseProducer<String> createFirehoseSinkFromStaticConfig() {
 /*
  *
com.amazonaws.services.kinesisanalytics.flink.connectors.config.ProducerConfigConstants
  * lists of all of the properties that firehose sink can be configured with.
  */

 Properties outputProperties = new Properties();
 outputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);

 FlinkKinesisFirehoseProducer<String> sink = new
FlinkKinesisFirehoseProducer<>(outputStreamName, new SimpleStringSchema(),
outputProperties);
 ProducerConfigConstants config = new ProducerConfigConstants();
 return sink;
}

private static FlinkKinesisFirehoseProducer<String>
createFirehoseSinkFromApplicationProperties() throws IOException {
 /*
  *
com.amazonaws.services.kinesisanalytics.flink.connectors.config.ProducerConfigConstants
  * lists of all of the properties that firehose sink can be configured with.
  */

 Map<String, Properties> applicationProperties =
KinesisAnalyticsRuntime.getApplicationProperties();
 FlinkKinesisFirehoseProducer<String> sink = new
FlinkKinesisFirehoseProducer<>(outputStreamName, new SimpleStringSchema(),
   applicationProperties.get("ProducerConfigProperties"));
 return sink;
}

public static void main(String[] args) throws Exception {
 // set up the streaming execution environment
 final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();


        /* if you would like to use runtime configuration properties, uncomment the lines
below
        * DataStream<String> input = createSourceFromApplicationProperties(env);
        */

 DataStream<String> input = createSourceFromStaticConfig(env);

 // Kinesis Firehose sink
```

```
  input.addSink(createFirehoseSinkFromStaticConfig());

  // If you would like to use runtime configuration properties, uncomment the lines below
  // input.addSink(createFirehoseSinkFromApplicationProperties());

  env.execute("Flink Streaming Java API Skeleton");
 }
}
```

有关如何使用 Kinesis Data Firehose 接收器的完整教程，请参阅the section called “Kinesis Data Firehose” (p. 155).

在 Kinesis Data Analytics 中将异步 I/O 用于 Apache Flink

异步 I/O 操作符使用外部数据源（例如数据库）以丰富流数据。Kinesis Data Analytics 异步地丰富流事件，以便可以批处理请求以提高效率。

有关更多信息，请参阅 。异步 I/O中的Apache Flink 文档.

使用 Apache Flink 在 Kinesis Data Analytics Data Analytics 中使用操作符以使用DataStreamAPI

要在 Apache Flink 使用 Kinesis Data Analytics Data Analytics for Apache Flink 应用程序，您可以使用 Apache Flink操作者. Apache Flink 操作符将一个或多个数据流转换为新的数据流。新数据流包含来自原始数据流的修改的数据。Apache Flink 提供超过 25 个预构建的流处理操作符。有关更多信息，请参阅 。运算符中的Apache Flink 文档.

本主题包含下列部分：

转换操作符

以下是对 JSON 数据流的某个字段进行简单文本转换的示例。

该代码创建转换的数据流。新数据流具有与原始流相同的数据，并在 `TICKER` 字段内容后面附加“`Company`”字符串。

```
DataStream<ObjectNode> output = input.map(
    new MapFunction<ObjectNode, ObjectNode>() {
        @Override
        public ObjectNode map(ObjectNode value) throws Exception {
            return value.put("TICKER", value.get("TICKER").asText() + " Company");
        }
    }
);
```

聚合操作符

以下是一个聚合操作符示例。该代码创建聚合的数据流。该操作符创建一个 5 秒的滚动窗口，并返回窗口中具有相同 `TICKER` 值的记录的 `PRICE` 值之和。

```
DataStream<ObjectNode> output = input.keyBy(node -> node.get("TICKER").asText())
    .window(TumblingProcessingTimeWindows.of(Time.seconds(5)))
```

```
    .reduce((node1, node2) -> {
        double priceTotal = node1.get("PRICE").asDouble() + node2.get("PRICE").asDouble();
        node1.replace("PRICE", JsonNodeFactory.instance.numberNode(priceTotal));
    return node1;
});
```

有关使用操作符的完整代码示例，请参阅入门 (DataStreamAPI) (p. 76)。入门应用程序的源代码可在开始使用中的Java Kinesis Data Analytics GitHub存储库。

使用DataStreamAPI

Kinesis Data Analytics 使用以下时间戳跟踪事件：

- 处理时间：指的是执行相应操作的计算机的系统时间。
- 事件时间：指的是在生成设备上发生每个事件的时间。
- 摄取时间：指的是事件进入 Kinesis Data Analytics 服务的时间。

您可以使用 `setStreamTimeCharacteristic` 设置流环境使用的时间：

```
env.setStreamTimeCharacteristic(TimeCharacteristic.ProcessingTime);
env.setStreamTimeCharacteristic(TimeCharacteristic.IngestionTime);
env.setStreamTimeCharacteristic(TimeCharacteristic.EventTime);
```

有关时间戳的更多信息，请参阅事件时间中的Apache Flink 文档.

表 API

你的 Apache Flink 应用程序使用Apache Flink 表 API使用关系模型与流中的数据交互。您可以使用表 API 使用表源访问数据，然后使用表函数转换和筛选表数据。您可以使用 API 函数或 SQL 命令转换和筛选表格数据。

本节包含以下主题：

- 表 API 连接器 (p. 13)：这些组件在应用程序与外部数据源和目标之间移动数据。
- 表 API 时间属性 (p. 14)：本主题介绍 Kinesis Data Analytics 如何在使用表 API 时跟踪事件。

表 API 连接器

在 Apache Flink 编程模型中，连接器是应用程序用于从外部源读取或写入数据的组件，例如其他Amazon服务。

使用 Apache Flink 表 API，您可以使用以下类型的连接器：

- 表 API 来源 (p. 13)：您可以使用表 API 源连接器在你的`TableEnvironment`使用 API 调用或 SQL 查询。
- 表 API 接收器 (p. 14)：您可以使用 SQL 命令将表数据写入外部源，例如 Amazon MSK 主题或 Amazon S3 存储桶。

表 API 来源

您可以从数据流创建表源。以下代码根据亚马逊 MSK 主题创建一个表：

```
//create the table
    final FlinkKafkaConsumer<StockRecord> consumer = new
 FlinkKafkaConsumer<StockRecord>(kafkaTopic, new KafkaEventDeserializationSchema(),
 kafkaProperties);
    consumer.setStartFromEarliest();
    //Obtain stream
    DataStream<StockRecord> events = env.addSource(consumer);

    Table table = streamTableEnvironment.fromDataStream(events);
```

有关表源的更多信息，请参阅。表和连接器中的Apache Flink 文档.

表 API 接收器

要将表数据写入汇，请在 SQL 中创建接收器，然后在`StreamTableEnvironment`对象。

以下代码示例演示了如何将表数据写入 Amazon S3 接收器：

```
final String s3Sink = "CREATE TABLE sink_table (" +
    "event_time TIMESTAMP," +
    "ticker STRING," +
    "price DOUBLE," +
    "dt STRING," +
    "hr STRING" +
    ")" +
    " PARTITIONED BY (ticker,dt,hr)" +
    " WITH" +
    "(" +
    " 'connector' = 'filesystem'," +
    " 'path' = '" + s3Path + "'," +
    " 'format' = 'json'" +
    ") ";

    //send to s3
    streamTableEnvironment.executeSql(s3Sink);
    filteredTable.executeInsert("sink_table");
```

您可以使用`format`参数来控制 Kinesis Data Analytics 用什么格式将输出写入接收器。有关格式的信息，请参阅。格式中的Apache Flink 文档.

有关表接收器的更多信息，请参阅。表和连接器中的Apache Flink 文档.

用户定义的源和汇

你可以使用现有的 Apache Kafka 连接器向其他人发送数据或从其他人之间发送Amazon服务，例如 Amazon MSK 和 Amazon S3。要与其他数据源和目标进行交互，您可以定义自己的源和汇。有关更多信息，请参阅 。用户定义的源和汇中的Apache Flink 文档.

表 API 时间属性

数据流中的每条记录都有几个时间戳，用于定义与记录相关的事件发生时间：

- 事件时间：用户定义的时间戳，用于定义创建记录的事件发生的时间戳。
- 提取时间：应用程序从数据流中检索记录的时间。
- 处理时间：应用程序处理记录的时间。

当 Apache Flink 表 API 基于创纪录的时间创建窗口时，您可以使用setStreamTime特征方法。

有关将时间戳与表 API 配合使用的更多信息，请参阅。时间属性中的Apache Flink 文档.

将 Python 结合使用 Kinesis Data Analytics

Apache Flink 版本 1.13.2 包括对使用 Python 3.8 版本创建应用程序的支持，使用PyFlink库. 您可以通过执行以下操作使用 Python 创建 Kinesis Data Analytics 应用程序：

- 将 Python 应用程序代码创建为文本文件，并使用`main`方法。
- 将应用程序代码文件和任何 Python 或 Java 依赖项捆绑到 zip 文件中，然后将其上传到 Amazon S3 存储桶。
- 创建 Kinesis Data Analytics 应用程序，指定 Amazon S3 代码位置、应用程序属性和应用程序设置。

在较高层次上，Python 表 API 是围绕 Java 表 API 的封装器。有关 Python 表 API 的信息，请参阅Python 表 API 简介中的Apache Flink 文档.

为 Python 应用程序编程 Kinesis Data Analytics

您可以使用 Apache Flink Python 表 API 为 Python 应用程序编码 Kinesis Data Analytics 应用程序。Apache Flink 引擎将 Python 表 API 语句（在 Python 虚拟机中运行）转换为 Java 表 API 语句（在 Java VM 中运行）。

您可以通过执行以下操作来使用 Python 表 API：

- 创建对`StreamTableEnvironment`.
- Create`table`对源流数据中的对象执行查询`StreamTableEnvironment`引用。
- 对你的`table`用于创建输出表的对象。
- 使用`StatementSet`.

要开始在 Kinesis Data Analytics 中使用 Python 表 API，请参阅Amazon Kinesis Data Analytics for Python Flink for P (p. 103).

读取和写入流数据

要读取和写入流数据，请在表环境中执行 SQL 查询。

创建表

以下代码示例演示了创建 SQL 查询的用户定义函数。SQL 查询创建了一个与 Kinesis 流交互的表：

```
def create_table(table_name, stream_name, region, stream_initpos):
   return """ CREATE TABLE {0} (
                `record_id` VARCHAR(64) NOT NULL,
                `event_time` BIGINT NOT NULL,
                `record_number` BIGINT NOT NULL,
                `num_retries` BIGINT NOT NULL,
                `verified` BOOLEAN NOT NULL
              )
              PARTITIONED BY (record_id)
              WITH (
                'connector' = 'kinesis',
                'stream' = '{1}',
                'aws.region' = '{2}',
                'scan.stream.initpos' = '{3}',
```

```
                'sink.partitioner-field-delimiter' = ';',
                'sink.producer.collection-max-count' = '100',
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(table_name, stream_name, region, stream_initpos)
```

阅读流媒体数据

以下代码示例演示如何使用上述`CreateTable`对表环境引用进行 SQL 查询以读取数据：

```
   table_env.execute_sql(create_table(input_table, input_stream, input_region,
 stream_initpos))
```

编写流媒体数据

以下代码示例演示如何使用中的 SQL 查询`CreateTable`创建输出表引用的示例，以及如何使用`StatementSet`与表交互以将数据写入目标 Kinesis 流：

```
   table_result = table_env.execute_sql("INSERT INTO {0} SELECT * FROM {1}"
                        .format(output_table_name, input_table_name))
```

读取运行时属性

您可以使用运行时属性配置应用程序，而无需更改应用程序代码。

为应用程序指定应用程序属性的方法与使用适用于 Java 的 Kinesis Data Analytics 应用程序的方式相同。您可以通过以下方式指定运行时属性：

- 使用CreateApplicationaction.
- 使用UpdateApplicationaction.
- 使用控制台配置应用程序。

通过读取名为的 json 文件来检索代码中的应用程序属性`application_properties.json`Kinesis Data Analytics 运行时创建的。

以下代码示例演示如何从中读取应用程序属性`application_properties.json`file:

```
file_path = '/etc/flink/application_properties.json'
   if os.path.isfile(file_path):
       with open(file_path, 'r') as file:
           contents = file.read()
           properties = json.loads(contents)
```

以下用户定义的函数代码示例演示了从应用程序属性对象读取属性组：检索：

```
def property_map(properties, property_group_id):
   for prop in props:
       if prop["PropertyGroupId"] == property_group_id:
           return prop["PropertyMap"]
```

以下代码示例演示从上一个示例返回的属性组中读取名为 INPUT_STREAM_KEY 的属性：

```
input_stream = input_property_map[INPUT_STREAM_KEY]
```

创建应用程序的代码包

创建 Python 应用程序后，将代码文件和依赖关系捆绑到一个 zip 文件中。

你的 zip 文件必须包含一个 python 脚本`main`方法，并可以选择包含以下内容：

- 其他 Python 代码文件
- JAR 文件中用户定义的 Java 代码
- JAR 文件中的 Java 库

> Note
>
> 您的应用程序 zip 文件必须包含应用程序的所有依赖关系。你不能为你的应用程序引用来自其他来源的库。

创建 Python Kinesis Data Analytics 应用程序

指定代码文件

创建应用程序代码包后，您将其上传到 Amazon S3 存储桶。然后，您可以使用控制台或CreateApplicationaction.

当你使用CreateApplication操作时，您可以使用名为的特殊应用程序属性组在 zip 文件中指定代码文件和存档`kinesis.analytics.flink.run.options`. 您可以定义以下类型的文件：

- python：包含 Python 主方法的文本文件。
- jarfile：包含 Java 用户定义函数的 Java JAR 文件。
- pyFiles：包含应用程序要使用的资源的 Python 资源文件。
- PyR档案馆：包含应用程序的资源文件的 zip 文件。

有关 Apache Flink Python 代码文件类型的更多信息，请参阅命令行使用中的Apache Flink 文档.

> Note
>
> Kinesis Data Analytics 不支持`pyModule`、`pyExecutable`，或者`pyRequirements`文件类型。所有代码、要求和依赖项都必须在你的 zip 文件中。您无法使用 pip 指定要安装的依赖关系。

以下示例 json 代码段演示了如何在应用程序的 zip 文件中指定文件位置：

```
"ApplicationConfiguration": {
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "kinesis.analytics.flink.run.options",
          "PropertyMap": {
            "python": "MyApplication/main.py",
            "jarfile": "MyApplication/lib/myJarFile.jar",
            "pyFiles": "MyApplication/lib/myDependentFile.py",
            "pyArchives": "MyApplication/lib/myArchive.zip"
          }
        },
```

监控 Python Kinesis Data Analytics 应用程序

你使用你的应用程序CloudWatch登录以监控 Python Kinesis Data Analytics 应用程序。

Kinesis Data Analytics 记录 Python 应用程序的以下消息：

- 使用写入控制台的消息`print()`在应用程序的`main`方法。
- 使用用户定义的函数发送的消息`logging`程序包。以下代码示例演示了从用户定义的函数写入应用程序日志的操作：

```
import logging

@udf(input_types=[DataTypes.BIGINT()], result_type=DataTypes.BIGINT())
def doNothingUdf(i):
    logging.info("Got {} in the doNothingUdf".format(str(i)))
    return i
```

- 应用程序抛出的错误消息。

  如果应用程序在`main`函数，它将出现在应用程序的日志中。

  以下示例演示了从 Python 代码抛出的异常的日志条目：

```
2021-03-15 16:21:20.000   --------------------------- Python Process Started
 ---------------------------
2021-03-15 16:21:21.000   Traceback (most recent call last):
2021-03-15 16:21:21.000   "  File ""/tmp/flink-web-6118109b-1cd2-439c-9dcd-218874197fa9/
flink-web-upload/4390b233-75cb-4205-a532-441a2de83db3_code/PythonKinesisSink/
PythonUdfUndeclared.py"", line 101, in <module>"
2021-03-15 16:21:21.000       main()
2021-03-15 16:21:21.000   "  File ""/tmp/flink-web-6118109b-1cd2-439c-9dcd-218874197fa9/
flink-web-upload/4390b233-75cb-4205-a532-441a2de83db3_code/PythonKinesisSink/
PythonUdfUndeclared.py"", line 54, in main"
2021-03-15 16:21:21.000   "    table_env.register_function(""doNothingUdf"",
 doNothingUdf)"
2021-03-15 16:21:21.000   NameError: name 'doNothingUdf' is not defined
2021-03-15 16:21:21.000   --------------------------- Python Process Exited
 ---------------------------
2021-03-15 16:21:21.000   Run python process failed
2021-03-15 16:21:21.000   Error occurred when trying to start the job
```

  Note

  由于性能问题，我们建议您只在应用程序开发期间使用自定义日志消息。

使用查询日志CloudWatch见解

以下CloudWatchInsights 查询会在执行应用程序的主要功能时搜索 Python 入口点创建的日志：

```
fields @timestamp, message
| sort @timestamp asc
| filter logger like /PythonDriver/
| limit 1000
```

Apache Flink 的 Kinesis Data Analytics 中的运行时属性

您可以使用运行时属性配置应用程序，而无需重新编译应用程序代码。

本主题包含下列部分：

在控制台中使用运行时属性

您可以使用控制台在 Kinesis Data Analytics 应用程序中添加、更新或删除运行时属性。

Note

在 Kinesis Data Analytics 控制台中创建应用程序时，您无法添加运行时属性。

更新 Kinesis Data Analytics 应用程序的运行时属性

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics.
2. 选择 Kinesis Data Analytics 应用程序。选择 Application details (应用程序详细信息)。
3. 在应用程序页面上，选择 Configure (配置)。
4. 展开 Properties (属性) 部分。
5. 使用 Properties (属性) 部分中的控件，以键值对形式定义一个属性组。可以使用这些控件添加、更新或删除属性组和运行时属性。
6. 选择 Update（更新）。

在 CLI 中使用运行时属性

您可以使用 Amazon CLI 添加、更新或删除运行时属性。

本节包含为应用程序配置运行时属性的 API 操作的示例请求。有关如何将 JSON 文件用于 API 操作输入的信息，请参阅 Kinesis Data Analytics API 示例代码 (p. 341)。

Note

将以下示例中的示例账户 ID (*012345678901*) 替换为您的账户 ID。

在创建应用程序时添加运行时属性

`CreateApplication` 操作的以下示例请求在创建应用程序时添加两个运行时属性组（`ProducerConfigProperties` 和 `ConsumerConfigProperties`）：

```
{
    "ApplicationName": "MyApplication",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_13",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/KA-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "java-getting-started-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
```

```
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

在现有应用程序中添加和更新运行时属性

UpdateApplication 操作的以下示例请求为现有应用程序添加或更新运行时属性：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 2,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap" : {
            "flink.stream.initpos" : "LATEST",
            "aws.region" : "us-west-2",
            "AggregationEnabled" : "false"
          }
        },
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap" : {
            "aws.region" : "us-west-2"
          }
        }
      ]
    }
  }
}
```

Note

如果您使用的键在属性组中没有相应的运行时属性，Kinesis Data Analytics 将键值对添加为新属性。如果将键用于属性组中的现有运行时属性，Kinesis Data Analytics 将更新属性值。

删除运行时属性

UpdateApplication 操作的以下示例请求从现有应用程序中删除所有运行时属性和属性组：

```
{
```

```
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 3,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": []
    }
  }
}
```

Important

如果省略现有的属性组或属性组中的现有属性键，则会删除该属性组或属性。

在 Kinesis Data Analytics 应用程序中访问运行时属性

您可以使用静态 `KinesisAnalyticsRuntime.getApplicationProperties()` 方法在 Java 应用程序代码中检索运行时属性，该方法返回一个 `Map<String, Properties>` 对象。

以下 Java 代码示例检索应用程序的运行时属性：

```
Map<String, Properties> applicationProperties =
KinesisAnalyticsRuntime.getApplicationProperties();
```

您按如下方式检索一个属性组（作为 `Java.Util.Properties` 对象）：

```
Properties consumerProperties = applicationProperties.get("ConsumerConfigProperties");
```

您通常，您传入`Properties`对象，而无需检索各个属性。以下代码示例说明了如何传入从运行时属性中检索的 `Properties` 对象以创建 Flink 源：

```
private static FlinkKinesisProducer<String> createSinkFromApplicationProperties() throws
 IOException {
  Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
  FlinkKinesisProducer<String> sink = new FlinkKinesisProducer<String>(new
 SimpleStringSchema(),
    applicationProperties.get("ProducerConfigProperties"));

  sink.setDefaultStream(outputStreamName);
  sink.setDefaultPartition("0");
  return sink;
}
```

有关使用运行时属性的完整代码示例，请参阅入门 (DataStreamAPI) (p. 76)。入门应用程序的源代码可在以下位置找到开始使用中的Kinesis Data Analytics Java 示例 GitHub存储库。

在 Kinesis Data Analytics 中为 Apache Flink 实施容错功能

检查点是用于在适用于 Apache Flink 的 Amazon Kinesis Data Analytics 中实施容错功能的方法。一个检查点是up-to-date备份运行的应用程序，用于立即从意外的应用程序中断或故障转移中恢复。

有关 Apache Flink 应用程序中检查点的详细信息，请参阅检查点中的Apache Flink 文档.

快照 是手动创建和管理的应用程序状态备份。通过使用快照，您可以调用 `UpdateApplication` 以将应用程序还原到以前的状态。有关更多信息，请参阅 使用快照管理应用程序备份 (p. 24)。

如果为应用程序启用了检查点，该服务将创建应用程序数据备份，并在应用程序意外重新启动时加载该备份以提供容错功能。这些意外的应用程序重新启动可能是由意外的作业重新启动、实例故障等引起的。这会在这些重新启动期间为应用程序提供与无故障执行相同的语义。

如果为应用程序启用了快照并使用应用程序的ApplicationRestore配置，则该服务在应用程序更新期间或与服务相关的扩展或维护期间提供恰好一次处理语义。

在 Kinesis Data Analytics 中为 Apache Flink 配置检查点

您可以配置应用程序的检查点行为。您可以定义它是否永久保存检查点状态、将其状态保存到检查点的频率以及一个检查点操作结束到另一个检查点操作开始之间的最小间隔。

您可以使用 `CreateApplication` 或 `UpdateApplication` API 操作配置以下设置：

- `CheckpointingEnabled`— 指示是否在应用程序中启用检查点。
- `CheckpointInterval`— 包含检查点（持久性）操作之间的时间（以毫秒为单位）。
- `ConfigurationType`— 将该值设置为`DEFAULT`以使用默认检查点行为。将该值设置为 `CUSTOM` 以配置其他值。

  Note

  默认检查点行为如下所示：
  - CheckpointingEnabled：真的
  - CheckpointInterval：60000
  - MinPauseBetweenCheckpoints：5000

  如果ConfigurationType设置为`DEFAULT`，将使用前面的值，即使通过使用Amazon Command Line Interface，或者通过在应用程序代码中设置值。
- `MinPauseBetweenCheckpoints`— 从一个检查点操作结束到另一个检查点操作开始之间的最短时间（以毫秒为单位）。如果设置该值，则可以防止应用程序在检查点操作所花的时间超过 `CheckpointInterval` 时继续执行检查点操作。

检查点 API 示例

本节包含为应用程序配置检查点的 API 操作的示例请求。有关如何将 JSON 文件用于 API 操作输入的信息，请参阅 Kinesis Data Analytics API 示例代码 (p. 341)。

为新应用程序配置检查点

`CreateApplication` 操作的以下示例请求在您创建应用程序时配置检查点：

```
{
   "ApplicationName": "MyApplication",
   "RuntimeEnvironment":"FLINK-1_13",
   "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
   "ApplicationConfiguration": {
      "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
```

```
        }
      },
      "FlinkApplicationConfiguration": {
         "CheckpointConfiguration": {
            "CheckpointingEnabled": "true",
            "CheckpointInterval": 20000,
            "ConfigurationType": "CUSTOM",
            "MinPauseBetweenCheckpoints": 10000
         }
      }
}
```

为新应用程序禁用检查点

`CreateApplication` 操作的以下示例请求在您创建应用程序时禁用检查点：

```
{
   "ApplicationName": "MyApplication",
   "RuntimeEnvironment":"FLINK-1_13",
   "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
   "ApplicationConfiguration": {
      "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "FlinkApplicationConfiguration": {
         "CheckpointConfiguration": {
            "CheckpointingEnabled": "false"
         }
      }
}
```

为现有应用程序配置检查点

`UpdateApplication` 操作的以下示例请求为现有应用程序配置检查点：

```
{
   "ApplicationName": "MyApplication",
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "CheckpointingEnabledUpdate": true,
            "CheckpointIntervalUpdate": 20000,
            "ConfigurationTypeUpdate": "CUSTOM",
            "MinPauseBetweenCheckpointsUpdate": 10000
         }
      }
   }
}
```

为现有应用程序禁用检查点

`UpdateApplication` 操作的以下示例请求为现有应用程序禁用检查点：

```
{
```

```
   "ApplicationName": "MyApplication",
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "CheckpointingEnabledUpdate": false,
            "CheckpointIntervalUpdate": 20000,
            "ConfigurationTypeUpdate": "CUSTOM",
            "MinPauseBetweenCheckpointsUpdate": 10000
         }
      }
   }
}
```

使用快照管理应用程序备份

一个快照是 Apache Flink 的 Kinesis Data Analytics 实现保存点. 快照是用户或服务触发、创建和管理的应用程序状态备份。有关 Apache Flink 保存点的信息，请参阅保存点中的Apache Flink 文档. 通过使用快照，您可以从特定的应用程序状态快照中重新启动应用程序。

> **Note**
>
> 我们建议应用程序每天创建几次快照以使用正确的状态数据正确地重新启动。快照的正确频率取决于应用程序的业务逻辑。通过频繁拍摄快照，您可以恢复更新的数据，但会增加成本并需要更多系统资源。

在 Kinesis Data Analytics 中，您可以使用以下 API 操作管理快照：

- `CreateApplicationSnapshot`
- `DeleteApplicationSnapshot`
- `DescribeApplicationSnapshot`
- `ListApplicationSnapshots`

有关每个应用程序的快照数限制，请参阅配额 (p. 242)。如果应用程序达到快照限制，则手动创建快照将失败并出现`LimitExceededException`.

Kinesis Data Analytics 永远不会删除快照。您必须使用 `DeleteApplicationSnapshot` 操作手动删除快照。

要在启动应用程序时加载已保存的应用程序状态快照，请使用`ApplicationRestoreConfiguration`的参数`StartApplication`要么`UpdateApplication`action.

本主题包含下列部分：

- 自动创建快照 (p. 24)
- 从包含不兼容状态数据的快照中还原 (p. 25)
- 快照 API 示例 (p. 25)

自动创建快照

如果`SnapshotsEnabled`设置为`true`中的 ApplicationSnapshot配置对于应用程序，Kinesis Data Analytics 会在应用程序更新、缩放或停止时自动创建和使用快照，以提供精确一次的处理语义。

> **Note**
>
> 如果将 `ApplicationSnapshotConfiguration::SnapshotsEnabled` 设置为 `false`，将导致在应用程序更新期间丢失数据。

自动创建的快照具有以下特性：

- 快照由服务管理，但是您可以使用 ListApplication快照action. 自动创建的快照计入您的快照限制。
- 如果应用程序超出快照限制，手动创建的快照将失败，但是当应用程序更新、缩放或停止时，Kinesis Data Analytics 服务仍会成功创建快照。您必须使用手动删除快照 DeleteApplication快照手动创建更多快照之前的操作。

从包含不兼容状态数据的快照中还原

由于快照包含有关操作符的信息，因此，如果从自上一应用程序版本以来发生变化的操作符的快照中还原状态数据，则可能会出现意外的结果。如果尝试从与当前操作符不对应的快照中还原状态数据，应用程序将会发生故障。发生故障的应用程序将停滞在 `STOPPING` 或 `UPDATING` 状态。

要允许应用程序从包含不兼容状态数据的快照中还原，请将`AllowNonRestoredState`的参数FlinkRun配置到`true`使用UpdateApplicationaction.

从过时的快照中还原应用程序时，您将会看到以下行为：

- 添加了操作符：如果添加了新操作符，则保存点没有新操作符的状态数据。不会发生故障，也不需要设置 `AllowNonRestoredState`。
- 已删除操作员：如果删除了现有操作符，则保存点具有丢失的操作符的状态数据。除非 `AllowNonRestoredState` 设置为 `true`，否则，将会发生故障。
- 操作员已修改：如果进行了兼容的更改，例如将参数的类型更改为兼容的类型，则应用程序可以从过时的快照中还原。有关从快照还原的更多信息，请参阅保存点中的Apache Flink 文档. 可以从具有不同架构的快照中还原使用 Apache Flink 版本 1.8 或更高版本的应用程序。无法还原使用 Apache Flink 版本 1.6 的应用程序。

如果您需要恢复与现有存储点数据不兼容的应用程序，我们建议您通过设置`ApplicationRestoreType`的参数StartApplication采取行动`SKIP_RESTORE_FROM_SNAPSHOT`.

有关 Apache Flink 如何处理不兼容的状态数据的更多信息，请参阅状态架构发展中的Apache Flink 文档.

快照 API 示例

本节包含将快照与应用程序一起使用的 API 操作的示例请求。有关如何将 JSON 文件用于 API 操作输入的信息，请参阅 Kinesis Data Analytics API 示例代码 (p. 341)。

为应用程序启用快照

`UpdateApplication` 操作的以下示例请求为应用程序启用快照：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "ApplicationSnapshotConfigurationUpdate": {
         "SnapshotsEnabledUpdate": "true"
       }
    }
}
```

创建快照

`CreateApplicationSnapshot` 操作的以下示例请求创建当前应用程序状态的快照：

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MyCustomSnapshot"
}
```

列出应用程序的快照

ListApplicationSnapshots 操作的以下示例请求列出当前应用程序状态的前 50 个快照：

```
{
   "ApplicationName": "MyApplication",
   "Limit": 50
}
```

列出应用程序快照的详细信息

DescribeApplicationSnapshot 操作的以下示例请求列出特定应用程序快照的详细信息：

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MyCustomSnapshot"
}
```

删除快照

DeleteApplicationSnapshot 操作的以下示例请求删除以前保存的快照。你可以获取SnapshotCreationTimestamp使用任何一种值ListApplicationSnapshots要么DeleteApplicationSnapshot：

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MyCustomSnapshot",
   "SnapshotCreationTimestamp": 12345678901.0,
}
```

使用命名的快照重新启动应用程序

StartApplication 操作的以下示例请求使用特定快照中保存的状态启动应用程序：

```
{
   "ApplicationName": "MyApplication",
   "RunConfiguration": {
      "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_CUSTOM_SNAPSHOT",
         "SnapshotName": "MyCustomSnapshot"
      }
   }
}
```

使用最近的快照重新启动应用程序

StartApplication 操作的以下示例请求使用最近的快照启动应用程序：

```
{
   "ApplicationName": "MyApplication",
```

```
   "RunConfiguration": {
      "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
      }
   }
}
```

不使用快照重新启动应用程序

`StartApplication` 操作的以下示例请求启动应用程序而不加载应用程序状态，即使具有快照也是如此：

```
{
   "ApplicationName": "MyApplication",
   "RunConfiguration": {
      "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "SKIP_RESTORE_FROM_SNAPSHOT"
      }
   }
}
```

Apache Flink 的 Kinesis Data Analytics 中的应用程序扩展

您可以为 Apache Flink 配置 Amazon Kinesis Data Analytics parallel 执行任务和资源分配，以实施扩展。有关 Apache Flink 如何计划 parallel 任务实例的信息，请参阅并行执行中的Apache Flink 文档.

主题

配置应用程序 Parallelism 和 ParallelismPerKPU

您可以使用以下命令为 Kinesis Data Analytics 应用程序任务（例如从源代码读取或执行运算符）配置 parallel 行执行`ParallelismConfiguration`属性：

- `Parallelism`使用该属性设置默认 Apache Flink 应用程序并行度。所有操作符、源和接收器以该并行度执行，除非在应用程序代码中覆盖它们。默认值为 `1`，最大值为 `256`。
- `ParallelismPerKPU`使用该属性设置可以为应用程序的每个 Kinesis 处理单元 (KPU) 计划的 parallel 任务数。默认值为 `1`，最大值为 `8`。对于具有阻止操作（例如，I/O）的应用程序，较高的 `ParallelismPerKPU` 值导致完全使用 KPU 资源。

> **Note**
>
> `Parallelism` 的限制等于 `ParallelismPerKPU` 乘以 KPU 限制（默认值为 32）。可以请求增加限制以增加 KPU 限制。有关如何请求增加限制的说明，请参阅中的 “请求增加限制”。Service Quotas.

有关为特定运算符设置任务并行度的信息，请参阅设置并行度：运算符中的Apache Flink 文档.

分配 Kinesis 处理单元

Kinesis Data Analytics 将容量预置为 KPU。一个 KPU 可为您提供 1 个 vCPU 和 4 GB 内存。对于分配的每个 KPU，还提供了 50 GB 运行的应用程序存储。

Kinesis Data Analytics 计算运行应用程序所需的 KPU，使用`Parallelism`和`ParallelismPerKPU`属性，如下所示：

```
Allocated KPUs for the application = Parallelism/ParallelismPerKPU
```

Kinesis Data Analytics 可以快速为应用程序提供资源，以应对出现的吞吐量或处理活动高峰。在活动高峰过后，它逐渐从应用程序中删除资源。要禁止自动分配资源，请将 `AutoScalingEnabled` 值设置为 `false`，如后面的更新应用程序的并行度 (p. 28)中所述。

应用程序的默认 KPU 限制为 32 个。有关如何请求增加该限制的说明，请参阅中的 “请求增加限制”。Service Quotas.

更新应用程序的并行度

本节包含设置应用程序并行度的 API 操作的示例请求。有关如何将请求块与 API 操作一起使用的更多示例和说明，请参阅Kinesis Data Analytics API 示例代码 (p. 341)。

`CreateApplication` 操作的以下示例请求在您创建应用程序时设置并行度：

```
{
   "ApplicationName": "string",
   "RuntimeEnvironment":"FLINK-1_13",
   "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
   "ApplicationConfiguration": {
      "ApplicationCodeConfiguration":{
      "CodeContent":{
         "S3ContentLocation":{
            "BucketARN":"arn:aws:s3:::mybucket",
            "FileKey":"myflink.jar",
            "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
            }
         },
      "CodeContentType":"ZIPFILE"
   },
      "FlinkApplicationConfiguration": {
         "ParallelismConfiguration": {
            "AutoScalingEnabled": "true",
            "ConfigurationType": "CUSTOM",
            "Parallelism": 4,
            "ParallelismPerKPU": 4
         }
      }
   }
}
```

`UpdateApplication` 操作的以下示例请求为现有的应用程序设置并行度：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "ParallelismConfigurationUpdate": {
```

```
            "AutoScalingEnabledUpdate": "true",
            "ConfigurationTypeUpdate": "CUSTOM",
            "ParallelismPerKPUUpdate": 4,
            "ParallelismUpdate": 4
         }
      }
   }
}
```

`UpdateApplication` 操作的以下示例请求为现有的应用程序禁用并行度：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "ParallelismConfigurationUpdate": {
            "AutoScalingEnabledUpdate": "false"
         }
      }
   }
}
```

自动扩展

Kinesis Data Analytics 可以灵活地扩展应用程序的并行度，以适应源数据的数据吞吐量，以及大多数情况下的操作符复杂性。Kinesis Data Analytics 可以监控应用程序的资源 (CPU) 使用情况，并相应地弹性地扩展应用程序的并行度。

- 当 CPU 使用率保持在 75% 或以上 15 分钟时，您的应用程序可以扩展（增加并行度）。
- 当 CPU 使用率在六个小时内保持在 10% 以下时，应用程序将缩小（减少并行度）。

Kinesis Data Analytics 不会减少你的应用`CurrentParallelism`价值低于你的应用程序`Parallelism`设置。

当 Kinesis Data Analytics 服务正在扩展您的应用程序时，它将在`AUTOSCALING`状态。您可以使用DescribeApplication要么ListApplications行动。当服务正在扩展您的应用程序时，您唯一可以使用的有效 API 操作是StopApplication使用`Force`将参数设置为`true`.

您可以使用 `AutoScalingEnabled` 属性（`FlinkApplicationConfiguration` 的一部分）启用或禁用自动扩展行为。您的AmazonKinesis Data Analytics 预置的 KPU 将被收费，Kinesis Data Analytics 预置的 KPU 取决于应用程序的`parallelism`和`parallelismPerKPU`设置。活动高峰将会增加您的 Kinesis Data Analytics 成本。

有关定价的信息，请参阅 Amazon Kinesis Data Analytics 定价。

请注意有关应用程序扩展的以下内容：

- 默认情况下，将会启用自动扩展。
- 缩放不适用于 Studio 笔记本。但是，如果将 Studio 笔记本作为持久状态的应用程序部署，则扩展将应用于已部署的应用程序。
- 应用程序的默认限制为 32 个 KPU。有关更多信息，请参阅 配额 (p. 242)。
- 在自动扩展更新应用程序并行度时，应用程序将会发生停机。为了避免这种停机，请执行以下操作：
  - 禁用自动扩展
  - 配置应用程序`parallelism`和`parallelismPerKPU`使用UpdateApplicationaction. 有关设置应用程序的并行度设置的更多信息，请参阅下面的the section called “更新应用程序的并行度” (p. 28)。

- 定期监控应用程序的资源使用情况，以验证应用程序是否具有适合其工作负载的并行度设置。有关监控分配资源使用情况的信息，请参阅the section called “指标与维度” (p. 214)。

MaxParalism 注意事项

- AutoScale 逻辑将防止将 Flink 作业扩展为可能会干扰作业和操作员的并行度`maxParallelism`. 例如，如果一个简单的作业只有来源和来源所有的汇`maxParallelism`16 和`sink`有 8 个，我们不会自动将作业扩展到 8 以上。
- 如果`maxParallelism`未为作业设置，Flink 将默认设置为 128。因此，如果您认为作业需要以高于 128 的并行度运行，则必须为应用程序设置该数字。
- 如果你希望看到你的工作自动缩放但没有看到它，请确保`maxParallelism`值允许它。

有关更多信息，请参阅Apache Flink 的增强监控和自动缩放

有关示例，请参阅 。kda flink-app 自动缩放.

使用标记

本节介绍如何将密钥值元数据标签添加到 Kinesis Data Analytics 应用程序。这些标签可用于以下目的：

- 确定各 Kinesis Data Analytics 应用程序的账单。有关更多信息，请参阅 。使用成本分配标签中的Billing and Cost Management 指南.
- 根据标签控制对应用程序资源的访问。有关更多信息，请参阅 。使用标签控制访问中的Amazon Identity and Access Management用户指南.
- 用户定义的目的。您可以根据用户标签定义应用程序的功能。

请注意与标记相关的以下信息：

- 应用程序标签的最大数量包括系统标签。用户定义的应用程序标签的最大数量为 50。
- 如果某项操作包含的标签列表存在重复的 `Key` 值，服务将提示 `InvalidArgumentException`。

本主题包含下列部分：

创建应用程序时添加标签

您可以使用在创建应用程序时添加标签`tags`的参数CreateApplicationaction.

以下示例请求显示了 `CreateApplication` 请求的 `Tags` 节点：

```
"Tags": [
    {
        "Key": "Key1",
        "Value": "Value1"
    },
    {
```

```
        "Key": "Key2",
        "Value": "Value2"
    }
]
```

为现有应用程序添加或更新标签

您可以使用将标签添加到应用程序中TagResourceaction. 您无法使用将标签添加到应用程序中。UpdateApplicationaction.

要更新现有标签，可添加一个与现有标签的键相同的标签。

针对 `TagResource` 操作的以下示例请求可添加新标签或更新现有标签：

```
{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "NewTagKey",
         "Value": "NewTagValue"
      },
      {
         "Key": "ExistingKeyOfTagToUpdate",
         "Value": "NewValueForExistingTag"
      }
   ]
}
```

列出应用程序的标签

要列出现有的标签，您可以使用ListTagsForResourceaction.

针对 `ListTagsForResource` 操作的以下示例请求可列出应用程序的标签：

```
{
   "ResourceARN": "arn:aws:kinesisanalytics:us-west-2:012345678901:application/
MyApplication"
}
```

从应用程序删除标签

要从应用程序中删除标签，您可以使用UntagResourceaction.

针对 `UntagResource` 操作的以下示例请求可从应用程序中删除标签：

```
{
   "ResourceARN": "arn:aws:kinesisanalytics:us-west-2:012345678901:application/
MyApplication",
   "TagKeys": [ "KeyOfFirstTagToRemove", "KeyOfSecondTagToRemove" ]
}
```

使用CloudFormation使用 Kinesis Data Analytics

以下练习演示如何启动通过以下方式创建的 Flink 应用程序：Amazon CloudFormation在同一堆栈中使用 Lambda 函数。

开始前的准备工作

在开始本练习之前，请按照以下步骤操作使用创建 Flink 应用程序Amazon CloudFormation
在AWS::KinesisAnalytics** 应用程序.

编写 Lambda 函数

要在创建或更新后启动 Flink 应用程序，我们使用 kinesisanalyticsv2启动申请API。该呼叫将由Amazon CloudFormationFlink 应用程序创建之后的事件。我们将在本练习后面讨论如何设置堆栈以触发 Lambda 函数，但首先我们将重点放在 Lambda 函数声明及其代码上。我们使用Python3.8本例中的运行时间。

```
StartApplicationLambda:
    Type: AWS::Lambda::Function
    DependsOn: StartApplicationLambdaRole
    Properties:
      Description: Starts an application when invoked.
      Runtime: python3.8
      Role: !GetAtt StartApplicationLambdaRole.Arn
      Handler: index.lambda_handler
      Timeout: 30
      Code:
        ZipFile: |
          import logging
          import cfnresponse
          import boto3

          logger = logging.getLogger()
          logger.setLevel(logging.INFO)

          def lambda_handler(event, context):
            logger.info('Incoming CFN event {}'.format(event))

            try:
              application_name = event['ResourceProperties']['ApplicationName']

              # filter out events other than Create or Update,
              # you can also omit Update in order to start an application on Create only.
              if event['RequestType'] not in ["Create", "Update"]:
                logger.info('No-op for Application {} because CFN RequestType {} is
 filtered'.format(application_name, event['RequestType']))
                cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

                return

              # use kinesisanalyticsv2 API to start an application.
              client_kda = boto3.client('kinesisanalyticsv2',
 region_name=event['ResourceProperties']['Region'])

              # get application status.
              describe_response =
 client_kda.describe_application(ApplicationName=application_name)
              application_status = describe_response['ApplicationDetail']
['ApplicationStatus']

              # an application can be started from 'READY' status only.
              if application_status != 'READY':
                logger.info('No-op for Application {} because ApplicationStatus {} is
 filtered'.format(application_name, application_status))
                cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

                return
```

```
            # create RunConfiguration.
            run_configuration = {
              'ApplicationRestoreConfiguration': {
                'ApplicationRestoreType': 'RESTORE_FROM_LATEST_SNAPSHOT',
              }
            }

            logger.info('RunConfiguration for Application {}:
 {}'.format(application_name, run_configuration))

            # this call doesn't wait for an application to transfer to 'RUNNING' state.
            client_kda.start_application(ApplicationName=application_name,
 RunConfiguration=run_configuration)

            logger.info('Started Application: {}'.format(application_name))
            cfnresponse.send(event, context, cfnresponse.SUCCESS, {})
          except Exception as err:
            logger.error(err)
            cfnresponse.send(event,context, cfnresponse.FAILED, {"Data": str(err)})
```

在上述代码中，Lambda 将处理传入Amazon CloudFormation事件，除此之外的所有内容`Create`和`Update`，获取应用程序状态并在状态为时启动它`READY`. 要获取应用程序状态，您需要创建 Lambda 角色，如下所示：

创建 Lambda 角色

您可以为 Lambda 创建一个角色，以便成功与应用程序 “交谈” 并编写日志。此角色将使用默认托管策略，但是您可能希望使用自定义策略缩小其范围。

```
StartApplicationLambdaRole:
    Type: AWS::IAM::Role
    DependsOn: TestFlinkApplication
    Properties:
      Description: A role for lambda to use while interacting with an application.
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - lambda.amazonaws.com
            Action:
              - sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonKinesisAnalyticsFullAccess
        - arn:aws:iam::aws:policy/CloudWatchLogsFullAccess
      Path: /
```

请注意，Lambda 资源将在同一堆栈中创建 Flink 应用程序之后创建，因为它们依赖于它。

调用 Lambda 函数

现在剩下的就是调用 Lambda 函数。这可以使用自定义资源.

```
StartApplicationLambdaInvoke:
    Description: Invokes StartApplicationLambda to start an application.
    Type: AWS::CloudFormation::CustomResource
    DependsOn: StartApplicationLambda
    Version: "1.0"
    Properties:
```

```
      ServiceToken: !GetAtt StartApplicationLambda.Arn
      Region: !Ref AWS::Region
      ApplicationName: !Ref TestFlinkApplication
```

这就是使用 Lambda 启动 Flink 应用程序所需的全部。现在，您已准备好创建自己的堆栈，或者使用下面的完整示例来了解所有这些步骤在实践中是如何运作的。

完整示例

以下示例是上述步骤的稍微扩展版本，其中包括一个额外的`RunConfiguration`通过调整模板参数. 这是一个可供你尝试的工作堆栈。请务必阅读随附的说明：

Stack.yaml

```
Description: 'KinesisAnalyticsV2 CloudFormation Test Application'
Parameters:
  ApplicationRestoreType:
    Description: ApplicationRestoreConfiguration option, can be SKIP_RESTORE_FROM_SNAPSHOT,
 RESTORE_FROM_LATEST_SNAPSHOT or RESTORE_FROM_CUSTOM_SNAPSHOT.
    Type: String
    Default: SKIP_RESTORE_FROM_SNAPSHOT
    AllowedValues: [ SKIP_RESTORE_FROM_SNAPSHOT, RESTORE_FROM_LATEST_SNAPSHOT,
 RESTORE_FROM_CUSTOM_SNAPSHOT ]
  SnapshotName:
    Description: ApplicationRestoreConfiguration option, name of a snapshot to restore to,
 used with RESTORE_FROM_CUSTOM_SNAPSHOT ApplicationRestoreType.
    Type: String
    Default: ''
  AllowNonRestoredState:
    Description: FlinkRunConfiguration option, can be true or false.
    Default: true
    Type: String
    AllowedValues: [ true, false ]
  CodeContentBucketArn:
    Description: ARN of a bucket with application code.
    Type: String
  CodeContentFileKey:
    Description: A jar filename with an application code inside a bucket.
    Type: String
Conditions:
  IsSnapshotNameEmpty: !Equals [ !Ref SnapshotName, '' ]
Resources:
  TestServiceExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - kinesisanalytics.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonKinesisFullAccess
        - arn:aws:iam::aws:policy/AmazonS3FullAccess
      Path: /
  InputKinesisStream:
    Type: AWS::Kinesis::Stream
    Properties:
      ShardCount: 1
  OutputKinesisStream:
    Type: AWS::Kinesis::Stream
```

```
  Properties:
    ShardCount: 1
TestFlinkApplication:
  Type: 'AWS::KinesisAnalyticsV2::Application'
  Properties:
    ApplicationName: 'CFNTestFlinkApplication'
    ApplicationDescription: 'Test Flink Application'
    RuntimeEnvironment: 'FLINK-1_13'
    ServiceExecutionRole: !GetAtt TestServiceExecutionRole.Arn
    ApplicationConfiguration:
      EnvironmentProperties:
        PropertyGroups:
          - PropertyGroupId: 'KinesisStreams'
            PropertyMap:
              INPUT_STREAM_NAME: !Ref InputKinesisStream
              OUTPUT_STREAM_NAME: !Ref OutputKinesisStream
              AWS_REGION: !Ref AWS::Region
      FlinkApplicationConfiguration:
        CheckpointConfiguration:
          ConfigurationType: 'CUSTOM'
          CheckpointingEnabled: True
          CheckpointInterval: 1500
          MinPauseBetweenCheckpoints: 500
        MonitoringConfiguration:
          ConfigurationType: 'CUSTOM'
          MetricsLevel: 'APPLICATION'
          LogLevel: 'INFO'
        ParallelismConfiguration:
          ConfigurationType: 'CUSTOM'
          Parallelism: 1
          ParallelismPerKPU: 1
          AutoScalingEnabled: True
      ApplicationSnapshotConfiguration:
        SnapshotsEnabled: True
      ApplicationCodeConfiguration:
        CodeContent:
          S3ContentLocation:
            BucketARN: !Ref CodeContentBucketArn
            FileKey: !Ref CodeContentFileKey
        CodeContentType: 'ZIPFILE'
StartApplicationLambdaRole:
  Type: AWS::IAM::Role
  DependsOn: TestFlinkApplication
  Properties:
    Description: A role for lambda to use while interacting with an application.
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - lambda.amazonaws.com
          Action:
            - sts:AssumeRole
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/AmazonKinesisAnalyticsFullAccess
      - arn:aws:iam::aws:policy/CloudWatchLogsFullAccess
    Path: /
StartApplicationLambda:
  Type: AWS::Lambda::Function
  DependsOn: StartApplicationLambdaRole
  Properties:
    Description: Starts an application when invoked.
    Runtime: python3.8
    Role: !GetAtt StartApplicationLambdaRole.Arn
    Handler: index.lambda_handler
```

```
      Timeout: 30
      Code:
        ZipFile: |
          import logging
          import cfnresponse
          import boto3

          logger = logging.getLogger()
          logger.setLevel(logging.INFO)

          def lambda_handler(event, context):
            logger.info('Incoming CFN event {}'.format(event))

            try:
              application_name = event['ResourceProperties']['ApplicationName']

              # filter out events other than Create or Update,
              # you can also omit Update in order to start an application on Create only.
              if event['RequestType'] not in ["Create", "Update"]:
                logger.info('No-op for Application {} because CFN RequestType {} is
 filtered'.format(application_name, event['RequestType']))
                cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

                return

              # use kinesisanalyticsv2 API to start an application.
              client_kda = boto3.client('kinesisanalyticsv2',
 region_name=event['ResourceProperties']['Region'])

              # get application status.
              describe_response =
 client_kda.describe_application(ApplicationName=application_name)
              application_status = describe_response['ApplicationDetail']
['ApplicationStatus']

              # an application can be started from 'READY' status only.
              if application_status != 'READY':
                logger.info('No-op for Application {} because ApplicationStatus {} is
 filtered'.format(application_name, application_status))
                cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

                return

              # create RunConfiguration from passed parameters.
              run_configuration = {
                'FlinkRunConfiguration': {
                  'AllowNonRestoredState': event['ResourceProperties']
['AllowNonRestoredState'] == 'true'
                },
                'ApplicationRestoreConfiguration': {
                  'ApplicationRestoreType': event['ResourceProperties']
['ApplicationRestoreType'],
                }
              }

              # add SnapshotName to RunConfiguration if specified.
              if event['ResourceProperties']['SnapshotName'] != '':
                run_configuration['ApplicationRestoreConfiguration']['SnapshotName'] =
 event['ResourceProperties']['SnapshotName']

              logger.info('RunConfiguration for Application {}:
 {}'.format(application_name, run_configuration))

              # this call doesn't wait for an application to transfer to 'RUNNING' state.
              client_kda.start_application(ApplicationName=application_name,
 RunConfiguration=run_configuration)
```

```
            logger.info('Started Application: {}'.format(application_name))
            cfnresponse.send(event, context, cfnresponse.SUCCESS, {})
          except Exception as err:
            logger.error(err)
            cfnresponse.send(event,context, cfnresponse.FAILED, {"Data": str(err)})
  StartApplicationLambdaInvoke:
    Description: Invokes StartApplicationLambda to start an application.
    Type: AWS::CloudFormation::CustomResource
    DependsOn: StartApplicationLambda
    Version: "1.0"
    Properties:
      ServiceToken: !GetAtt StartApplicationLambda.Arn
      Region: !Ref AWS::Region
      ApplicationName: !Ref TestFlinkApplication
      ApplicationRestoreType: !Ref ApplicationRestoreType
      SnapshotName: !Ref SnapshotName
      AllowNonRestoredState: !Ref AllowNonRestoredState
```

同样，您可能希望调整 Lambda 的角色以及应用程序本身的角色。

在创建上面的堆栈之前，别忘记指定你的参数。

参数 .json

```
[
  {
    "ParameterKey": "CodeContentBucketArn",
    "ParameterValue": "YOUR_BUCKET_ARN"
  },
  {
    "ParameterKey": "CodeContentFileKey",
    "ParameterValue": "YOUR_JAR"
  },
  {
    "ParameterKey": "ApplicationRestoreType",
    "ParameterValue": "SKIP_RESTORE_FROM_SNAPSHOT"
  },
  {
    "ParameterKey": "AllowNonRestoredState",
    "ParameterValue": "true"
  }
]
```

Replace`YOUR_BUCKET_ARN`和`YOUR_JAR`符合你的具体要求。你可以按照此操作指南创建 Amazon S3 存储桶和应用程序 jar。

现在创建堆栈（用你选择的区域替换 YOUR_REGION，例如 us-east-1）：

```
aws cloudformation create-stack --region YOUR_REGION --template-body "file://stack.yaml"
 --parameters "file://parameters.json" --stack-name "TestKDAStack" --capabilities
 CAPABILITY_NAMED_IAM
```

现在，您可转至https://console.aws.amazon.com/cloudformation然后查看进度。创建之后，你应该在中看到你的 Flink 应用程序`Starting`状态。开始可能需要几分钟的时间`Running`.

有关更多信息，请参阅下列内容：

- 四种方法可以检索任何Amazon使用服务属性Amazon CloudFormation（第 1 部分，共 3 部分）.
- 演练：查找 Amazon 系统映像 ID.

将 Apache Flink 控制面板与 Amazon Kinesis Data Analytics 结合使用

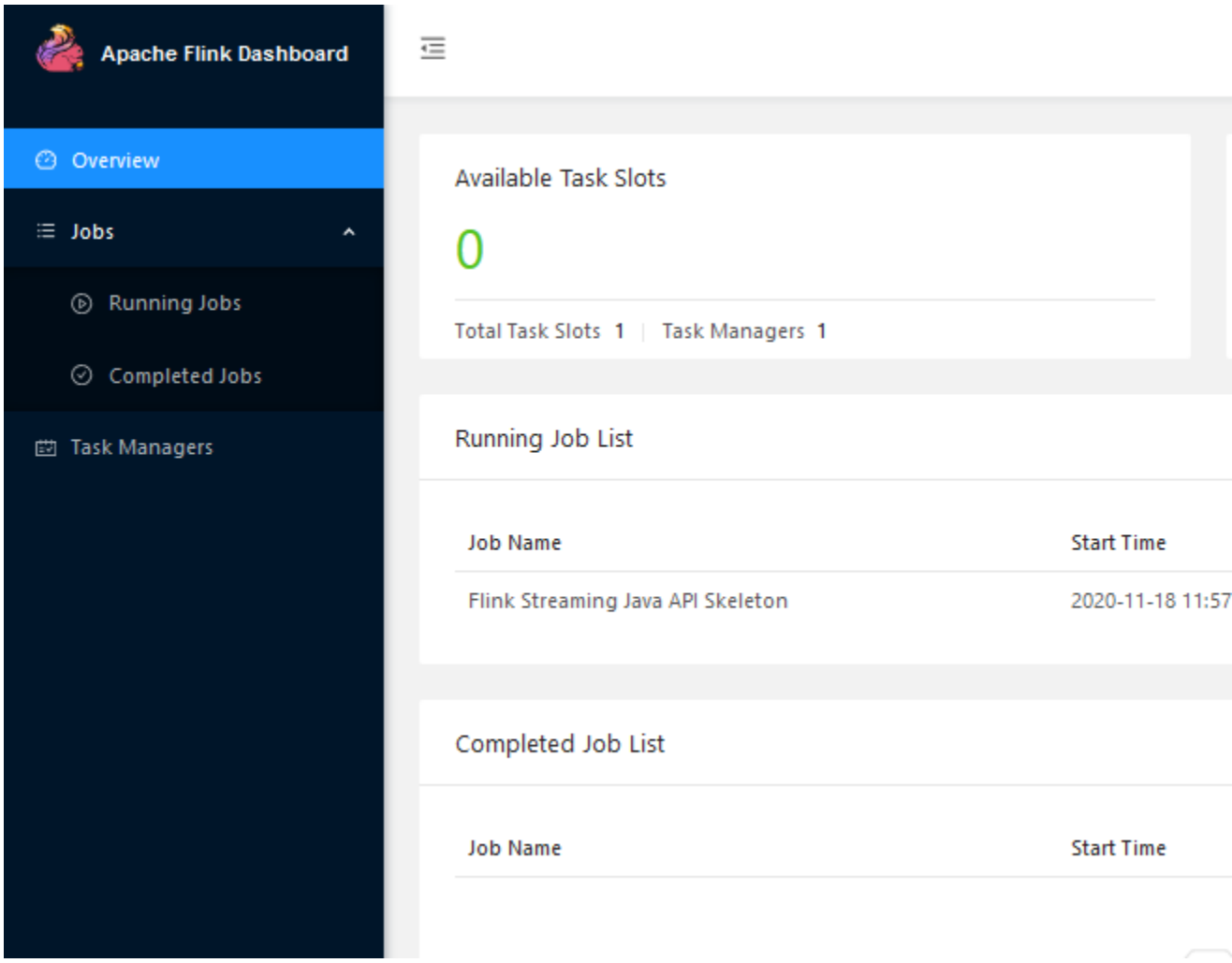
您可以使用应用程序的 Apache Flink 控制面板来监控 Kinesis Data Analytics 应用程序的运行状况。应用程序的控制面板显示以下信息：

- 正在使用的资源，包括任务管理器和任务插槽。
- 有关作业的信息，包括正在运行、已完成、取消和失败的作业。

有关 Apache Flink 任务管理器、任务插槽和作业的信息，请参阅Apache Flink 架构在 Apache Flink 网站上。

请注意以下关于将 Apache Flink 仪表板与 Kinesis Data Analytics 应用程序结合使用的事项：

- 用于 Kinesis Data Analytics 应用程序的 Apache Flink 控制面板是只读的。您无法使用 Apache Flink 控制面板对 Kinesis Data Analytics 应用程序进行更改。
- Apache Flink 控制面板与微软互联网资源管理器不兼容。

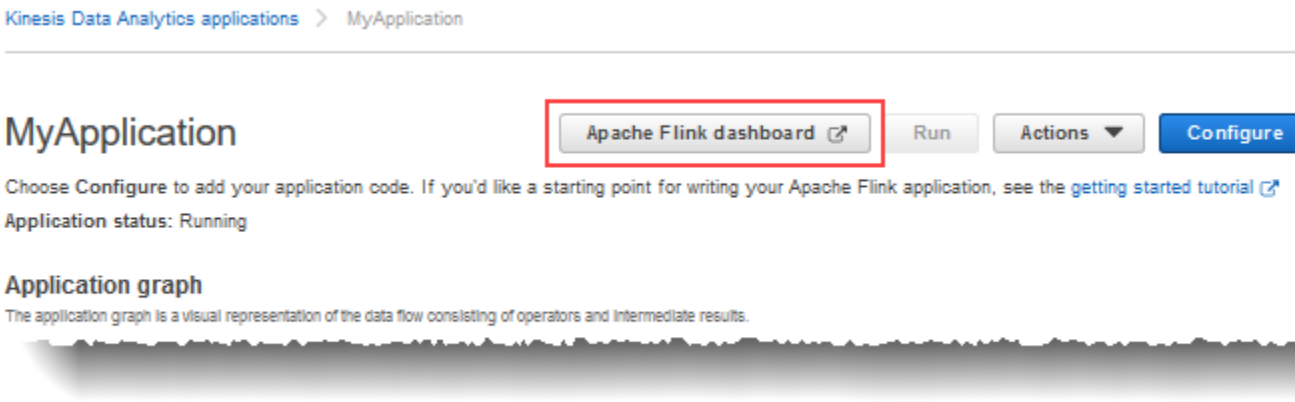


访问应用程序的 Apache Flink 控制面板

您可以通过 Kinesis Data Analytics 控制台或使用 CLI 请求安全 URL 终端节点来访问应用程序的 Apache Flink 控制面板。

使用 Kinesis Data Analytics 控制台访问应用程序的 Apache Flink 控制面板

要从控制台访问应用程序的 Apache Flink 控制面板，请选择Apache Flink 控制面板在应用程序页面上。



Note

当您从 Kinesis Data Analytics 控制台打开仪表板时，控制台生成的 URL 将有效期为 12 小时。

使用 Kinesis Data Analytics CLI 访问应用程序的 Apache Flink 控制面板

您可以使用 Kinesis Data Analytics CLI 生成 URL 以访问应用程序控制面板。您生成的 URL 在指定的时间内有效。

Note

如果你在三分钟内没有访问生成的 URL，它将不再有效。

您可以使用 CreateApplicationPresignedUrlaction. 为操作指定以下参数：

- 应用程序名称
- URL 的有效时间（以秒为单位）
- 你指定`FLINK_DASHBOARD_URL`作为 URL 类型。

将 Kinesis Data Analytics

适用于 Kinesis Data Analytics 的 Studio 笔记本允许您以交互方式实时查询数据流，并使用标准 SQL、Python 和 Scala 轻松构建和运行流处理应用程序。只需点击几下Amazon管理控制台，您可以启动无服务器笔记本来查询数据流并在几秒钟内获得结果。

笔记本是一种基于 Web 的开发环境。有了笔记本电脑，您将获得简单的交互式开发体验与 Apache Flink 提供的高级功能相结合。Studio 笔记本电脑使用支持的笔记本电脑Apache Zeppelin，以及用途Apache Flink作为流处理引擎。Studio 笔记本无缝结合了这些技术，使各种技能的开发人员都能访问数据流的高级分析。

Apache Zeppelin 为您的 Studio 笔记本提供了一整套分析工具，包括以下工具：

- 数据可视化
- 将数据导出到文件
- 控制输出格式以简化分析

要开始使用 Kinesis Data Analytics 和 Apache Zeppelin，请参阅创建 Studio 笔记本教程 (p. 48). 有关 Apache Zeppelin 的更多信息，请参阅Apache Zeppelin 文档.

有了笔记本，你可以使用 Apache Flink 为查询建模表 API 和 SQL在 SQL、Python 或 Scala 中，或者DataStream API在Sala。只需点击几下，您就可以将 Studio 笔记本升级为持续运行的非交互式 Kinesis Data Analytics 流处理应用程序，用于生产工作负载。

本主题包含下列部分：

创建 Studio 笔记本

Studio 笔记本包含以 SQL、Python 或 Scala 编写的查询或程序，这些查询或程序在流数据上运行并返回分析结果。您可以使用控制台或 CLI 创建应用程序，并提供用于分析数据源中数据的查询。

您的应用程序包含以下组件：

- 数据源，例如 Amazon MSK 集群、Kinesis 数据流或 Amazon S3 存储桶。
- Amazon Glue 数据库。此数据库包含存储数据源和目标架构和终端节点的表。有关更多信息，请参阅 使用 Amazon Glue (p. 47)。
- 应用程序代码。您的代码实现了您的分析查询或程序。
- 您的应用程序设置和运行时属性。有关应用程序设置和运行时属性的信息，请参阅Apache Link 应用程序开发者指南：
  - 应用程序并行和扩展：您可以使用应用程序的 Parallelism 设置来控制应用程序可以同时执行的查询数。如果您的查询具有多个执行路径，例如在以下情况下，也可以利用提高的并行度：
    - 处理 Kinesis 数据流的多个分片时
    - 使用对数据进行分区时`KeyBy`运算符。
    - 使用多个窗口运算符时

    有关应用程序扩展的更多信息，请参阅。Kinesis Data Analytics.
  - 日志记录和监控：有关应用程序日志记录和监视的信息，请参阅。Amazon Kinesis Data Analytics.
  - 您的应用程序使用检查点和保存点来实现容错。Studio 笔记本默认情况下不启用检查点和保存点。

您可以使用以下任一方法创建 Studio 笔记本Amazon Web Services Management Console或者Amazon CLI.

在控制台创建应用程序时，为您提供了以下选项：

- 在 Amazon MSK 控制台中，选择您的集群，然后选择实时处理数据.
- 在 Kinesis Data Streams 控制台中，选择您的数据流，然后在应用程序选项卡选择实时处理数据.
- 在 Kinesis Data Analytics 控制台中，选择工作室选项卡，然后选择创建 Studio.

有关如何创建 Stdudio 笔记本的教程Amazon Web Services Management Console或者Amazon CLI，请参阅教程：在 Kinesis Data Analytics 中创建 Studio 笔记本 (p. 48).

有关更高级的 Studio 笔记本解决方案的示例，请参阅Amazon Kinesis Data Analytics.

流数据的交互式分析

您可以使用由 Apache Zeppelin 提供支持的无服务器笔记本与流数据进行交互。您的笔记本可以有多个笔记，每个注释可以有一个或多个段落，您可以在其中编写代码。

以下示例 SQL 查询显示了如何从数据源检索数据：

```
%flink.ssql(type=update)
select * from stock;
```

有关 Flink 流式处理 SQL 查询的更多示例，请参阅示例和教程s (p. 48)以下权限，以及。查询中的Apache Flink 文档.

你可以在 Studio 笔记本中使用 Flink SQL 查询来查询流数据。你也可以使用 Python（表 API）和 Scala（表和数据流 API）来编写程序以交互方式查询你的流数据。您可以查看查询或程序的结果，在几秒钟内更新它们，然后重新运行它们以查看更新的结果。

Flink 口译员

您可以指定 Kinesis Data Analytics 使用哪种语言来运行应用程序，方法是使用翻译员. 您可以在 Kinesis Data Analytics 中使用以下解释器：

名称	类	描述
%flink	FlinkInterpreter	Creates ExecutionEnvironment/ StreamExecutionEnvironment/ BatchTableEnvironment/ StreamTableEnvironment and provides a Scala environment
%flink.pyflink	PyFlinkInterpreter	Provides a python environment
%flink.ipyflink	IPyFlinkInterpreter	Provides an ipython environment
%flink.ssql	FlinkStreamSqlInterpreter	Provides a stream sql environment
%flink.bsql	FlinkBatchSqlInterpreter	Provides a batch sql environment

有关 Flink 解释器的更多信息，请参阅Apache Zeppelin 的 Flink 口译员.

如果您将`%flink.pyflink`要么`%flink.ipyflink`作为您的口译员，您需要使用`ZeppelinContext`在笔记本中可视化结果。

有关 PyFlink 具体示例，请参阅。使用 Kinesis Data Analytics 工作室和 Python 以交互方式查询数据流.

Apache Flink 表环境变量

Apache Zeppelin 使用环境变量提供对表环境资源的访问。

您可以使用以下变量访问 Scala 表环境资源：

变量	资源
`senv`	`StreamingTableEnvironment`
`benv`	`ExecutionEnvironment`
`stenv`	`StreamTableEnvironment #######`
`btenv`	`BatchTableEnvironment #######`
`stenv_2`	`StreamTableEnvironment flink plink p`
`btenv_2`	`BatchTableEnvironment fllink Plink`

您可以使用以下变量访问 Python 表环境资源：

变量	资源
`s_env`	`StreamingTableEnvironment`
`b_env`	`ExecutionEnvironment`
`st_env`	`StreamTableEnvironment #######`
`bt_env`	`BatchTableEnvironment #######`
`st_env_2`	`StreamTableEnvironment fllink Plink`

变量	资源
bt_env_2	BatchTableEnvironment fllink Plink

有关使用表环境的更多信息，请参阅。创建 TableEnvironment中的Apache Flink 文档.

作为具有持久状态的应用程序进行部署

您可以构建您的代码并将其导出到 Amazon S3。您可以将您在笔记中编写的代码提升到持续运行的流处理应用程序。在 Kinesis Data Analytics 上运行 Apache Flink 应用程序有两种模式：借助 Studio 笔记本，您可以以交互方式开发代码、实时查看代码结果并在注释中进行可视化。部署注释以在流模式下运行后，Kinesis Data Analytics 会为您创建一个应用程序，该应用程序可以持续运行，从源读取数据，写入目标，保持长时间运行的应用程序状态，并根据源流的吞吐量自动缩放。

Note

要将应用程序代码导出到的 S3 存储桶必须与 Studio 笔记本位于同一区域。

只有满足以下条件的 Studio 笔记本才能部署笔记：

- 段落必须按顺序排列。在部署您的应用程序时，注释中的所有段落将按顺序执行（left-to-right、 top-to-bottom），因为它们出现在你的笔记中。您可以通过选择查看此订单运行所有段落在你的笔记中。
- 你的代码是 Python 和 SQL 的组合，或者是 Scala 和 SQL 的组合。我们目前不支持 Python 和 Scala 一起使用 deploy-as-application.
- 你的笔记应该只有以下口译员：`%flink`、`%flink.ssql`、`%flink.pyflink`、`%flink.ipyflink`、`%md`.
- 的用途Zeppelin宾语`z`不支持 。不返回任何内容的方法除了记录警告之外不会执行任何操作。其他方法会引发 Python 异常或无法在 Scala 中编译。
- 注释必须产生一个 Apache Flink 作业。
- 备注动态表单不支持作为应用程序部署。
- %md (Markown) 段落在部署为应用程序时将被跳过，因为这些段落应该包含不适合作为生成的应用程序的一部分运行的人类可读文档。
- 在部署为应用程序时，将跳过因在 Zeppelin 中运行而禁用的段落。例如，即使禁用的段落使用了不兼容的解释器，`%flink.ipyflink`在附注中`%flink and %flink.ssql`解释器，在将注释作为应用程序部署时将被跳过，并且不会导致错误。
- 必须至少有一个段落包含源代码（Flink SQL， PyFlink 或 Flink Scala），它已启用运行以使应用程序部署成功。
- 在段落中的解释器指令中设置并行度（例如`%flink.ssql(parallelism=32)`）将在从注释部署的应用程序中被忽略。而是可以更新已部署的应用程序Amazon Web Services Management Console、Amazon Command Line Interface要么Amazon用于更改并行度的 API 和/或 ParallelismPer根据应用程序所需的并行级别设置 KPU，或者您可以为已部署的应用程序启用自动缩放。

斯卡拉/Python 标准

- 你的 Scala 或 Python 代码不能使用`BatchExecutionEnvironment`要么`BatchTableEnvironment(benv、btenv、btenv_2fala；b_env、bt_env、bt_env_2for` Python)。
- 在你的 Scala 或 Python 代码中，使用闪烁规划师(`senv`、`stenvfala；s_env`、`st_env`对于 Python）而不是较旧的 “Flink” 计划器（`stenv_2`对于Sala，`st_env_2`for Python)。Apache Flink 项目建议在生产用例中使用 Blink 计划器，这是齐柏林飞艇和 Flink 中的默认计划器。

- 你的 Python 段落一定不能使用shell 调用/赋值使用 ! 要么IPython 魔法命令像`%timeit`要么`%conda`在注释中打算作为应用程序进行部署。
- 你不能使用Scala案例类作为传递给高阶数据流运算符的函数的参数`map`和`filter`. 有关 Scala 案例类的信息，请参阅案例类在 Scala 文档中。

Sala 标准

- 不允许使用简单的 SELECT 语句，因为没有任何地方可以等同于段落的输出部分可以传送数据。
- 在任何给定段落中，DDL 语句 (`USE`、`CREATE`、`ALTER`、`DROP`、`SET`、`RESET`) 必须位于 DML (`INSERT`) 语句。这是因为段落中的 DML 语句必须作为单个 Flink 作业一起提交。
- 最多只能有一个段落包含 DML 语句。这是因为，对于 deploy-as-application 功能，我们只支持向 Flink 提交一份作业。

有关更多信息以及示例，请参阅Amazon Kinesis Data Analytics.

Studio Notebooks I

当你通过创建 Studio 笔记本时，Kinesis Data Analytics 会为你创建 IAM 角色Amazon Web Services Management Console. 它还将允许以下访问权限的策略与该角色关联：

服务	访问
CloudWatch 日志	List
Amazon EC2	List
Amazon Glue	读、写。
Kinesis Data Analytics	Read
Kinesis Data Analytics V2	Read
Amazon S3	读、写。

连接器和依赖项

连接器支持您读取和写入数据。Kinesis Data Analytics 将三个默认连接器与 Studio 笔记本捆绑在一起。您还可以使用自定义连接器。有关连接器的更多信息，请参阅。表和 SQL 连接器在 Apache Flink 文档中。

默认连接器

如果您将Amazon Web Services Management Console要创建 Studio 笔记本，Kinesis Data Analytics 默认包含以下自定义连接器：`flink-sql-connector-flink`、`flink-connector-kafka_2.12`和`aws-msk-iam-auth`. 要通过控制台创建 Studio 笔记本而不使用这些自定义连接器，请选择使用自定义设置创建选项。然后，当你到达配置页面上，清除两个连接器旁边的复选框。

如果您将CreateApplication用于创建 Studio 笔记本的 API`flink-sql-connector-flink`和`flink-connector-kafka`默认情况下不包括连接器。要添加它们，请将它们指定为`MavenRefernce`中的`CustomArtifactsConfiguration`数据类型，如以下示例所示。

这些区域有：aws-msk-iam-auth连接器是与 Amazon MSK 配合使用的连接器，它包含自动通过 IAM 进行身份验证的功能。

Note

以下示例中显示的连接器版本是我们支持的唯一版本。

```
For the Kinesis connector:

"CustomArtifactsConfiguration": [{
   "ArtifactType": "DEPENDENCY_JAR",
   "MavenReference": {
      "GroupId": "org.apache.flink",
      "ArtifactId": "flink-sql-connector-kinesis-2.12",
      "Version": "1.13.2"
   }
}]

For the Apache MSK connector:

"CustomArtifactsConfiguration": [{
   "ArtifactType": "DEPENDENCY_JAR",
   "MavenReference": {
      "GroupId": "software.amazon.msk",
      "ArtifactId": "aws-msk-iam-auth",
      "Version": "1.1.0"
   }
}]

For the Apache Kafka connector:

"CustomArtifactsConfiguration": [{
   "ArtifactType": "DEPENDENCY_JAR",
   "MavenReference": {
      "GroupId": "org.apache.flink",
      "ArtifactId": "flink-connector-kafka_2.12",
      "Version": "1.13.2"
   }
}]
```

要将这些连接器添加到现有笔记本电脑，请使用UpdateApplicationAPI 操作并将它们指定为MavenReference中的CustomArtifactsConfigurationUpdate数据类型。

Note

您可以设置。failOnError对于 trueflink-sql-connector-kinesis表 API 中的连接器。

依赖关系和自定义连接器

使用Amazon Web Services Management Console要向 Studio 笔记本添加依赖项或自定义连接器，请执行以下步骤：

1. 将您的自定义连接器文件上载到 Amazon S3。
2. 在Amazon Web Services Management Console，选择自定义创建用于创建 Studio 笔记本的选项。
3. 遵循 Studio 笔记本创建工作流程，直至进入配置Step。
4. 在自定义连接器部分中，选择。添加自定义连接器.
5. 指定依赖项的 Amazon S3 位置。
6. 选择Save changes（保存更改）。

使用创建新 Studio 笔记本时添加依赖项 JAR 或自定义连接器CreateApplicationAPI，指定依赖项 JAR 或自定义连接器的 Amazon S3 位置`CustomArtifactsConfiguration`数据类型。要向现有 Studio 笔记本添加依赖项或自定义连接器，请调用UpdateApplicationAPI 操作并指定依赖项 JAR 或自定义连接器的 Amazon S3 位置`CustomArtifactsConfigurationUpdate`数据类型。

Note

当包含依赖项或自定义连接器时，还必须包括其所有未捆绑在其中的传递依赖项。

用户定义的函数

用户定义函数 (UDF) 是扩展点，允许您调用常用逻辑或自定义逻辑，这些逻辑无法在查询中以其他方式表示。你可以使用 Python 或 JVM 语言（如 Java 或 Scala）在 Studio 笔记本中的段落中实现你的 UDF。您还可以将包含以 JVM 语言实现的 UDF 的外部 JAR 文件添加到 Studio 笔记本中。

要使用控制台将 UDF JAR 文件添加到 Studio 笔记本中，请执行以下步骤：

1. 将您的 UDF JAR 文件上载到 Amazon S3。
2. 在Amazon Web Services Management Console，选择自定义创建用于创建 Studio 笔记本的选项。
3. 遵循 Studio 笔记本创建工作流程，直至进入配置Step。
4. 在用户定义的函数部分中，选择。添加用户定义的函数.
5. 指定 JAR 文件或具有 UDF 实现的 ZIP 文件的 Amazon S3 位置。
6. 选择Save changes（保存更改）。

在创建新 Stdio 笔记本时添加 UDF JARCreateApplicationAPI，指定 JAR 位置在`CustomArtifactConfiguration`数据类型。要将 UDF JAR 添加到现有的 Studio 笔记本中，请调用UpdateApplicationAPI 操作并在中指定 JAR 位置`CustomArtifactsConfigurationUpdate`数据类型。或者，您也可以使用Amazon Web Services Management Console将 UDF JAR 文件添加到您的 Studio 笔记本中。

启用检查点

您可以使用环境设置启用检查点功能。有关检查点操作的信息，请参阅。容错能力中的Kinesis Data Analytics.

设置检查点间隔

以下 Scala 代码示例将应用程序的检查点间隔设置为一分钟：

```
// start a checkpoint every 1 minute
stenv.enableCheckpointing(60000)
```

以下 Python 代码示例将应用程序的检查点间隔设置为一分钟：

```
st_env.get_config().get_configuration().set_string(
    "execution.checkpointing.interval", "1min"
)
```

设置检查点类型

以下 Scala 代码示例将应用程序的检查点模式设置为`EXACTLY_ONCE`（默认）：

```
// set mode to exactly-once (this is the default)
stenv.getCheckpointConfig.setCheckpointingMode(CheckpointingMode.EXACTLY_ONCE)
```

以下 Python 代码示例将您的应用程序的检查点模式设置为EXACTLY_ONCE（默认）：

```
st_env.get_config().get_configuration().set_string(
    "execution.checkpointing.mode", "EXACTLY_ONCE"
)
```

使用 Amazon Glue

您的 Studio 笔记本存储并从中获取有关其数据源和接收器的信息Amazon Glue. 创建 Studio 笔记本时，您可以指定Amazon Glue包含您的连接信息的数据库。访问数据源和汇时，您可以指定Amazon Glue数据库中包含的表。您的Amazon Glue表格提供对Amazon Glue用于定义数据源和目标的位置、架构和参数的连接。

Studio 笔记本使用表属性来存储特定于应用程序的数据。有关更多信息，请参阅 表属性 (p. 47)。

有关如何设置Amazon Glue用于 Studio 笔记本的连接、数据库和表，请参阅创建Amazon Glue数据库 (p. 49)中的创建 Studio 笔记本教程 (p. 48)教程。

表属性

除了数据字段，您的Amazon Glue表格使用表属性向 Studio 笔记本提供其他信息。Kinesis Data AnalyticsAmazon Glue表属性：

- 使用 Apache Flink 时间值 (p. 47)：这些属性定义了 Kinesis Data Analytics 如何发出 Apache Flink 内部数据处理时间值。
- 使用 Flink 连接器和格式属性 (p. 48)：这些属性提供有关数据流的信息。

将属性添加到Amazon Glue表中，执行以下操作：

1. 登录 Amazon Web Services Management Console，然后打开 Amazon Glue 控制台，网址为：https://console.aws.amazon.com/glue/。
2. 从表的列表中，选择您的应用程序用于存储其数据连接信息的表。选择操作、编辑表格详细信息.
3. UNDER表属性，输入。**kinesisanalytics.proctime**为了密钥和**user_action_time**为了值.

使用 Apache Flink 时间值

Apache Flink 提供了描述流处理事件何时发生的时间值，例如处理时间和事件时间. 要在应用程序输出中包含这些值，请在Amazon Glue表，告诉 Kinesis Data Analytics 运行时将这些值发送到指定的字段中。

您在表属性中使用的键和值如下所示：

时间戳类型	密钥	值
处理时间	kinesisanalytics.proctime	The column name that Amazon Glue will use to expose the value. This column name does not correspond to an existing table column.

时间戳类型	密钥	值
事件时间	kinesisanalytics.rowtime	The column name that Amazon Glue will use to expose the value. This column name corresponds to an existing table column.
	kinesisanalytics.warmark。*column_name*.watermarks	The watermark interval in milliseconds

使用 Flink 连接器和格式属性

您可以使用以下命令向应用程序的 Flink 连接器提供有关数据源的信息Amazon Glue表属性。以下是 Kinesis Data Analytics 用于连接器的属性的一些示例：

连接器类型	密钥	值
Kafka	`format`	The format used to deserialize and serialize Kafka messages, e.g. `json` or `csv`.
	`scan.startup.mode`	The startup mode for the Kafka consumer, e.g. `###ofset` or `timestamp`.
Kinesis	`format`	The format used to deserialize and serialize Kinesis data stream records, e.g. `json` or `csv`.
	`aws.region`	The Amazon region where the stream is defined.
S3（文件系统）	format	The format used to deserialize and serialize files, e.g. `json` or `csv`.
	`path`	The Amazon S3 path, e.g. `s3://mybucket/`.

有关除 Kinesis 和 Apache Kafka 之外的其他连接器的更多信息，请参阅连接器的文档。

示例和教程s

主题

- 教程：在 Kinesis Data Analytics 中创建 Studio 笔记本 (p. 48)
- 教程：部署为具有持久状态的应用程序 (p. 61)
- 示例 (p. 63)

教程：在 Kinesis Data Analytics 中创建 Studio 笔记本

以下教程演示了如何创建从 Kinesis 数据流或 Amazon MSK 集群读取数据的 Studio 笔记本。

本教程包含以下部分：

设置

确保你的Amazon CLI是 2 或更高版本。安装最新Amazon CLI，请参阅安装、更新和卸载Amazon CLI版本 2.

创建Amazon Glue数据库

你的 Studio 笔记本使用Amazon Glue有关亚马逊 MSK 数据源的元数据库。

创建Amazon Glue数据库

1. 打开Amazon Glue控制台https://console.aws.amazon.com/glue/.
2. 选择添加数据库. 在添加数据库窗口中，输入**`default`**为了数据库名称. 选择 Create（创建）。

后续步骤

通过本教程，您可以创建一个使用 Kinesis Data Streams 或 Amazon MSK 的 Studio 笔记本：

- Kinesis Data Streams (p. 49)：使用 Kinesis Data Streams，您可以快速创建使用 Kinesis 数据流作为源的应用程序。您只需要创建 Kinesis 数据流作为依赖资源。
- Amazon MSK (p. 53)：使用 Amazon MSK，您可以创建一个使用亚马逊 MSK 集群作为源的应用程序。您需要创建 Amazon VPC、Amazon EC2 客户端实例和 Amazon MSK 集群作为依赖资源。

使用 Kinesis Data Streams 创建 Studio 笔记本

本教程描述如何创建将 Kinesis Data Streams 用作源的 Studio 笔记本。

本教程包含以下部分：

设置

在创建 Studio 笔记本之前，请创建 Kinesis 数据流 (`ExampleInputStream`)。您的应用程序将此流用于应用程序源。

您可以使用 Amazon Kinesis 控制台或以下命令创建此流Amazon CLI命令。有关控制台说明，请参阅创建和更新数据流中的Amazon Kinesis Data Streams 开发人员指南. 命名流**`ExampleInputStream`**然后设置打开的分片数量到**`1`**.

创建流 (`ExampleInputStream`) 使用Amazon CLI，使用以下 Amazon Kinesis`create-stream` Amazon CLI命令。

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-east-1 \
--profile adminuser
```

创建一个 Amazon Glue 表

你的 Studio 笔记本使用Amazon Glue有关 Kinesis Data Streams 数据源的元数据库。

> Note
>
> 您可以先手动创建数据库，也可以让 Kinesis Data Analytics 在创建笔记本时为您创建数据库。同样，您可以按本节所述手动创建表，也可以在 Apache Zeppelin 的笔记本中使用 Kinesis Data Analytics 的创建表连接器代码通过 DDL 语句创建表。然后你可以办理登机手续Amazon Glue以确保表格已正确创建。

创建表

1. 登录 Amazon Web Services Management Console，然后打开 Amazon Glue 控制台，网址为：https://console.aws.amazon.com/glue/。
2. 如果您还没有Amazon Glue选择数据库数据库从左侧导航栏中。选择添加数据库. 在添加数据库窗口中，输入**`default`**为了数据库名称. 选择 Create（创建）。
3. 在左侧导航栏中，选择表. 在表页面上，选择添加表格、手动添加表.
4. 在设置表的属性页面，输入**`stock`**(对于)表名称. 确保选择以前创建的数据库。请选择 Next (下一步)。
5. 在添加数据存储页面上，选择Kinesis. 对于流名称，输入**`ExampleInputStream`**. 适用于Kinesis 源 URL，选择 Enter**`https://kinesis.us-east-1.amazonaws.com`**. 如果您复制并粘贴Kinesis 源 URL，请务必删除任何前导空格或尾随空格。请选择 Next (下一步)。
6. 在Classification页面上，选择JSON. 请选择 Next (下一步)。
7. 在定义架构页面上，选择添加列以添加列。添加具有以下属性的列：

列名称	数据类型
`#####`	**`###`**
`##`	**`double`**

   请选择 Next (下一步)。
8. 在下一页上，验证设置，然后选择Finish.
9. 从表列表中选择您新创建的表。
10. 选择编辑表然后用钥匙添加属性`kinesisanalytics.proctime`和价值`proctime`.
11. 选择 Apply（应用）。

使用 Kinesis Data Streams 创建 Studio 笔记本

既然您已经创建了应用程序使用的资源，就可以创建 Studio 笔记本。

要创建应用程序，您可以使用Amazon Web Services Management Console或者Amazon CLI.

- 使用创建 Studio 笔记本Amazon Web Services Management Console (p. 51)

- 使用创建 Studio 笔记本Amazon CLI (p. 51)

使用创建 Studio 笔记本Amazon Web Services Management Console

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics/home?region=us-east-1#/applications/dashboard.
2. 在Kinesis Data Analytics页面上，选择工作室选项卡。选择创建 Studio 笔记本.

   Note

   您还可以从 Amazon MSK 或 Kinesis Data Streams 控制台创建 Studio 笔记本，方法是选择输入的亚马逊 MSK 集群或 Kinesis 数据流，然后选择实时处理数据.
3. 在创建 Studio 笔记本页面上，提供以下信息：

   - Enter**`MyNotebook`**用于笔记本的名称。
   - 选择默认为了AmazonGlue Data.

   选择创建 Studio 笔记本.
4. 在MyNotebook页面上，选择运行. 等待状态显示正在运行. 笔记本电脑运行时需支付费用。

使用创建 Studio 笔记本Amazon CLI

使用创建 Studio 笔记本Amazon CLI中，执行以下操作：

1. 验证您的账户 ID。您需要此值才能创建应用程序。
2. 创建角色`arn:aws:iam::`*`AccountID`*`:role/ZeppelinRole`然后通过控制台向自动创建的角色添加以下权限。

   ```
   "kinesis:GetShardIterator",

   "kinesis:GetRecords",

   "kinesis:ListShards"
   ```
3. 创建以下内容的名为 `create.json` 的文件。将占位符值替换为您的信息。

   ```
   {
       "ApplicationName": "MyNotebook",
       "RuntimeEnvironment": "ZEPPELIN-FLINK-2_0",
       "ApplicationMode": "INTERACTIVE",
       "ServiceExecutionRole": "arn:aws:iam::AccountID:role/ZeppelinRole",
       "ApplicationConfiguration": {
           "ApplicationSnapshotConfiguration": {
               "SnapshotsEnabled": false
           },
           "ZeppelinApplicationConfiguration": {
               "CatalogConfiguration": {
                   "GlueDataCatalogConfiguration": {
                       "DatabaseARN": "arn:aws:glue:us-east-1:AccountID:database/default"
                   }
               }
           }
       }
   }
   ```
4. 运行以下命令以创建应用程序：

   ```
   aws kinesisanalyticsv2 create-application --cli-input-json file://create.json
   ```

5. 命令完成后，您会看到显示新 Studio 笔记本电脑详细信息的输出。下面是输出的一个示例。

```
{
    "ApplicationDetail": {
        "ApplicationARN": "arn:aws:kinesisanalytics:us-east-1:012345678901:application/
MyNotebook",
        "ApplicationName": "MyNotebook",
        "RuntimeEnvironment": "ZEPPELIN-FLINK-2_0",
        "ApplicationMode": "INTERACTIVE",
        "ServiceExecutionRole": "arn:aws:iam::012345678901:role/ZeppelinRole",
...
```

6. 运行以下命令以启动应用程序。将示例值替换为您的账户 ID。

```
aws kinesisanalyticsv2 start-application --application-arn arn:aws:kinesisanalytics:us-
east-1:012345678901:application/MyNotebook\
```

将数据发送到 Kinesis 数据流

要将测试数据发送到 Kinesis 数据流，请执行以下操作：

1. 打开Kinesis Data Fenesis in.
2. 选择使用创建 Cognito 用户CloudFormation.
3. 这些区域有：Amazon CloudFormation打开控制台并打开 Kinesis Data Generator 模板。请选择 Next (下一步)。
4. 在指定堆栈细节页面上，输入 Cognito 用户的用户名和密码。请选择 Next (下一步)。
5. 在配置堆栈选项页面上，选择下一步.
6. 在查看 Kinesis-Data 生成器-认知用户页面上，选择我承认Amazon CloudFormation可能会创建 IAM 资源。复选框。选择创建堆栈。
7. 等待Amazon CloudFormation创建堆栈完成。堆栈完成后，打开Kinesis-Data 生成器-认知用户堆叠在Amazon CloudFormation控制台，然后选择输出选项卡。打开列出的 URLKinesisDataGeneratorUrl输出值。
8. 在Amazon Kinesis Data Fenator页面上，使用您在步骤 4 中创建的凭证登录。
9. 在下一页上，提供以下值：

区域	**us-east-1**
流/交付流	**ExampleInput#**
每秒记录数	**1**

适用于录制模板中，粘贴以下代码：

```
{
    "ticker": "{{random.arrayElement(
        ["AMZN","MSFT","GOOG"]
    )}}",
    "price": {{random.number(
        {
            "min":10,
            "max":150
        }
    )}}
}
```

10. 选择发送数据.
11. 发生器将向 Kinesis Data Streams 发送数据流。

    完成下一部分时，让发电机保持运行状态。

测试 Studio 笔记本

在本节中，您可以使用 Studio 笔记本来查询 Kinesis 数据流中的数据。

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics/home?region=us-east-1#/applications/dashboard.
2. 在存储库的Kinesis Data Analytics页面上，选择Studio 笔记本选项卡。选择MyNotebook.
3. 在MyNotebook页面上，选择在 Apache Zeppelin 中打开.

   Apache Zeppelin 界面将在新的选项卡中打开。
4. 在欢迎使用 Zeppelin！页面上，选择Zeppelin Note.
5. 在Zeppelin Note页面上，在新备注中输入以下查询：

   ```
   %flink.ssql(type=update)
   select * from stock
   ```

   选择运行图标。

   短时间后，该笔记显示来自 Kinesis 数据流的数据。

要为应用程序打开 Apache Flink 控制面板以查看运营方面，请选择FLINK 作业. 有关 Flink 控制面板的更多信息，请参阅Apache Flink 控制面板中的Kinesis Data Analytics 开发者指南.

有关 Flink Streams SQL 查询的更多示例，请参阅查询中的Apache Flink 文档.

使用亚马逊 MSK 创建 Studio 笔记本

本教程描述如何创建将 Amazon MSK 集群用作源的 Studio 笔记本。

本教程包含以下部分：

设置

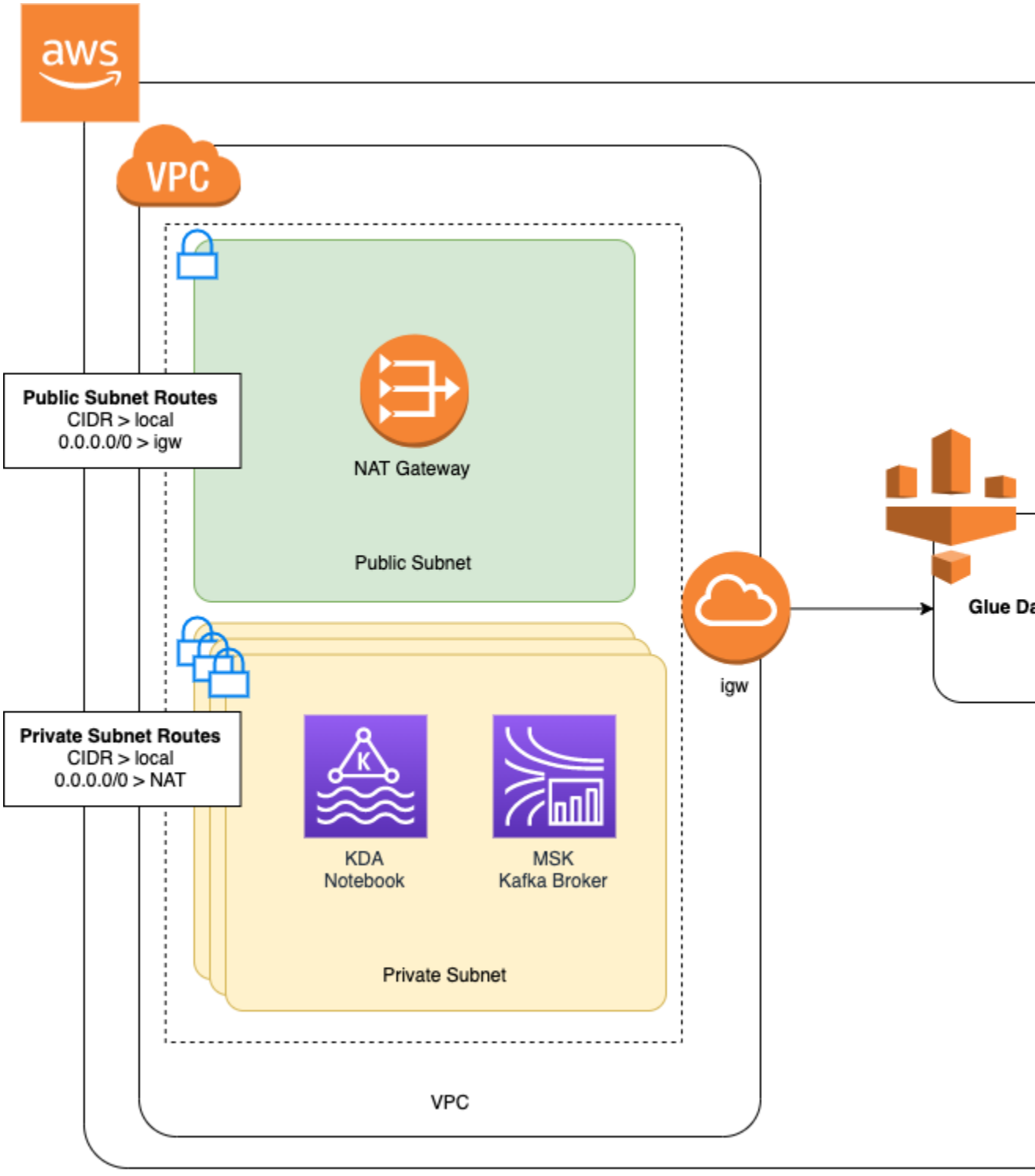
对于本教程，您需要一个允许明文访问的 Amazon MSK 群集。如果您还没有设置 Amazon MSK 集群，请按照开始使用亚马逊 MSK创建 Amazon VPC、Amazon MSK 集群、主题和 Amazon EC2 客户端实例的教程。

在遵循本教程时，请执行以下操作：

- In步骤 3: 创建亚马逊 MSK 集群，在步骤 4 中，更改`ClientBroker`价值来自`TLS`到**`PLAINTEXT`**.

将 NAT 网关添加到您的 VPC

如果您通过遵循开始使用亚马逊 MSK教程，或者如果您现有的 Amazon VPC 还没有用于其私有子网的 NAT 网关，则必须向您的 Amazon VPC 添加 NAT 网关。下图演示了架构。



要为您的 Amazon VPC 创建 NAT Gateway，请执行以下操作：

1. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
2. 选择NAT 网关从左侧导航栏中。
3. 在存储库的NAT 网关页面上，选择创建 NAT 网关.
4. 在存储库的创建 NAT 网关页面上，提供以下值：

姓名-可选的	**#####**
子网	AmazonKafkatTorialSubnet1
弹性 IP 分配 ID	Choose an available Elastic IP. If there are no Elastic IPs available, choose 分配弹性 IP, and then choose the Elasic IP that the console creates.

   选择创建 NAT 网关.
5. 在左侧导航栏上，选择路由表.
6. 选择 Create Route Table。
7. 在存储库的创建路由表页面上，提供以下信息：

   - 名称标签：**ZeppelinRouteTable**
   - VPC：选择你的 VPC（例如AmazonKafkatutorialVPC)。

   选择 Create（创建）。
8. 在路由表列表中，选择齐柏林路由表. 选择路由选项卡，然后选择编辑路线.
9. 在编辑路线页面上，选择添加路由.
10. 在适用于目的地输入**0.0.0.0/0**. 适用于目标，选择NAT 网关、齐柏林网关. 选择保存路由. 选择关闭。
11. 在路由表页面上，使用齐柏林路由表已选中，选择子网关联选项卡。选择编辑子网关联.
12. 在编辑子网关联页面上，选择AmazonKafkatutorialSubnet2和AmazonKafkatTorialSubnet3. 选择 Save（保存）。

创建Amazon Glue连接和表

你的 Studio 笔记本使用Amazon Glue有关亚马逊 MSK 数据源的元数据库。在本节中，您将创建Amazon Glue描述如何访问 Amazon MSK 集群的连接，以及Amazon Glue该表介绍了如何向客户端（例如 Studio 笔记本电脑）呈现数据源中的数据。

创建连接

1. 登录 Amazon Web Services Management Console，然后打开 Amazon Glue 控制台，网址为：https://console.aws.amazon.com/glue/。
2. 如果您还没有Amazon Glue数据库，选择数据库从左侧导航栏中。选择添加数据库. 在添加数据库窗口中，输入**default**为了数据库名称. 选择 Create（创建）。
3. 选择连接从左侧导航栏中。选择添加连接.
4. 在添加连接在窗口中，提供以下值：

   - 适用于连接名称输入**ZeppelinConnection**.
   - 对于 Connection type (连接类型)，选择 Kafka。
   - 适用于Kafka 引导启动服务器 URL，为您的集群提供引导代理字符串。您可以从 MSK 控制台或通过输入以下 CLI 命令获取引导经纪商：

```
aws kafka get-bootstrap-brokers --region us-east-1 --cluster-arn ClusterArn
```

- 取消选中需要 SSL 连接复选框。

选择 Next（下一步）。

5. 在VPC页面上，提供以下值：
   - 适用于VPC中，选择您的 VPC 的名称（例如 AmazonKafkatutorialVPC。）
   - 适用于子网，选择AmazonKafkatutorialSubnet2.
   - 适用于安全组中，选择所有可用的组。

   选择 Next（下一步）。
6. 在连接属性/连接访问页面上，选择Finish.

创建表

Note

您可以按照以下步骤所述手动创建表，也可以在 Apache Zeppelin 的笔记本中使用 Kinesis Data Analytics 的创建表连接器代码通过 DDL 语句创建表。然后你可以办理登机手续Amazon Glue以确保表格已正确创建。

1. 在左侧导航栏中，选择表. 在表页面上，选择添加表、手动添加表.
2. 在设置表的属性页面，输入**stock**(对于)表名称. 确保选择以前创建的数据库。选择 Next（下一步）。
3. 在添加数据存储页面上，选择Kafka. 对于主题名称，输入你的主题名称（例如AmazonKafkat教程主题)。适用于Connection，选择齐柏林连接.
4. 在Classification页面上，选择JSON. 选择 Next（下一步）。
5. 在定义架构在页面上，选择添加列以添加列。添加具有以下属性的列：

列名称	数据类型
***#####	***###
***##	**double**

   选择 Next（下一步）。
6. 在下一页上，验证您的设置，然后选择Finish.
7. 从表的列表中选择您新创建的表。
8. 选择编辑表然后用钥匙添加属性`kinesisanalytics.proctime`和价值`proctime`.
9. 选择 Apply（应用）。

使用亚马逊 MSK 创建 Studio 笔记本

既然您已经创建了应用程序使用的资源，就可以创建 Studio 笔记本。

您可以使用Amazon Web Services Management Console或者Amazon CLI.

- 使用创建 Studio 笔记本Amazon Web Services Management Console (p. 57)
- 使用创建 Studio 笔记本Amazon CLI (p. 51)

Note

您还可以通过选择现有群集然后选择从 Amazon MSK 控制台创建 Studio 笔记本实时处理数据.

使用创建 Studio 笔记本Amazon Web Services Management Console

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics/home?region=us-east-1#/applications/dashboard.
2. 在Kinesis Data Analytics页面上，选择工作室选项卡。选择创建 Studio 笔记本.

   Note

   要从 Amazon MSK 或 Kinesis Data Streams 控制台创建 Studio 笔记本电脑，请选择输入的亚马逊 MSK 集群或 Kinesis 数据流，然后选择实时处理数据.
3. 在创建 Studio 笔记本页面上，提供以下信息：

   - Enter**`MyNotebook`**为了Studio 笔记本名称.
   - 选择默认为了AmazonGlue 数据库.

   选择创建 Studio 笔记本.
4. 在MyNotebook页面上，选择配置选项卡。在联网部分，选择编辑.
5. 在为 myNotebook 编辑网络页面上，选择基于 Amazon MSK 集群的 VPC 配置. 为选择您的 Amazon MSK 集群亚马逊 MSK 集群. 选择 Save changes（保存更改）。
6. 在MyNotebook页面上，选择运行. 等待状态显示正在运行.

使用创建 Studio 笔记本Amazon CLI

使用创建 Studio 笔记本Amazon CLI中，执行以下操作：

1. 验证您是否具有以下信息。您需要使用这些值来创建您的应用程序。

   - 您的 账户 ID
   - 包含 Amazon MSK 集群的 Amazon VPC 的子网 ID 和安全组 ID。
2. 创建以下内容的名为 `create.json` 的文件。将占位符值替换为您的信息。

```
{
    "ApplicationName": "MyNotebook",
    "RuntimeEnvironment": "ZEPPELIN-FLINK-2_0",
    "ApplicationMode": "INTERACTIVE",
    "ServiceExecutionRole": "arn:aws:iam::AccountID:role/ZeppelinRole",
    "ApplicationConfiguration": {
        "ApplicationSnapshotConfiguration": {
            "SnapshotsEnabled": false
        },
        "VpcConfigurations": [
            {
                "SubnetIds": [
                    "SubnetID 1",
                    "SubnetID 2",
                    "SubnetID 3"
                ],
                "SecurityGroupIds": [
                    "VPC Security Group ID"
                ]
            }
        ],
        "ZeppelinApplicationConfiguration": {
```

```
            "CatalogConfiguration": {
                "GlueDataCatalogConfiguration": {
                    "DatabaseARN": "arn:aws:glue:us-east-1:AccountID:database/default"
                }
            }
        }
    }
}
```

3. 运行以下命令以创建您的应用程序：

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create.json
```

4. 当命令完时，您应看到类似于以下内容的输出，其中显示新 Studio 笔记本的详细信息：

```
{
    "ApplicationDetail": {
        "ApplicationARN": "arn:aws:kinesisanalytics:us-east-1:012345678901:application/
MyNotebook",
        "ApplicationName": "MyNotebook",
        "RuntimeEnvironment": "ZEPPELIN-FLINK-2_0",
        "ApplicationMode": "INTERACTIVE",
        "ServiceExecutionRole": "arn:aws:iam::012345678901:role/ZeppelinRole",
...
```

5. 运行以下命令以启动您的应用程序。将示例值替换为您的账户 ID。

```
aws kinesisanalyticsv2 start-application --application-arn arn:aws:kinesisanalytics:us-
east-1:012345678901:application/MyNotebook\
```

将数据发送到您的亚马逊 MSK 集群

在本节中，您在 Amazon EC2 客户端中运行 Python 脚本以将数据发送到您的 Amazon MSK 数据源。

1. Connect 到您的 Amazon EC2 客户端。
2. 运行以下命令安装 Python 版本 3、Pip 和 Kafka of Python 软件包，然后确认操作：

```
sudo yum install python37
curl -O https://bootstrap.pypa.io/get-pip.py
python3 get-pip.py --user
pip install kafka-python
```

3. 配置Amazon CLI在客户端计算机上输入以下命令：

```
aws configure
```

提供您的账户凭证，**us-east-1**(对于)region.

4. 创建以下内容的名为 stock.py 的文件。将示例值替换为亚马逊 MSK 集群的 Bootstrap Broker 字符串，如果主题不是，则更新主题名称AmazonKafkat教程主题：

```
from kafka import KafkaProducer
import json
import random
from datetime import datetime

BROKERS = "<<Bootstrap Broker List>>"
producer = KafkaProducer(
    bootstrap_servers=BROKERS,
```

```
    value_serializer=lambda v: json.dumps(v).encode('utf-8'),
    retry_backoff_ms=500,
    request_timeout_ms=20000,
    security_protocol='PLAINTEXT')


def getStock():
    data = {}
    now = datetime.now()
    str_now = now.strftime("%Y-%m-%d %H:%M:%S")
    data['EVENT_TIME'] = str_now
    data['ticker'] = random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV'])
    price = random.random() * 100
    data['price'] = round(price, 2)
    return data


while True:
    data =getStock()
    # print(data)
    try:
        future = producer.send("AWSKafkaTutorialTopic", value=data)
        producer.flush()
        record_metadata = future.get(timeout=10)
        print("sent event to Kafka! topic {} partition {} offset
 {}".format(record_metadata.topic, record_metadata.partition, record_metadata.offset))
    except Exception as e:
        print(e.with_traceback())
```

5. 使用以下命令运行脚本：

```
$ python3 stock.py
```

6. 在完成以下部分时，请将脚本保持运行状态。

测试 Studio 笔记本

在本节中，您可以使用 Studio 笔记本查询来自亚马逊 MSK 集群的数据。

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics/home?region=us-east-1#/applications/dashboard.
2. 在存储库的Kinesis Data Analytics页面上，选择Studio 笔记本选项卡。选择MyNotebook.
3. 在MyNotebook页面上，选择在 Apache Zeppelin 中打开.

   在新的选项卡中打开 Apache Zeppelin 界面。
4. 在欢迎使用 Zeppelin！页面上，选择Zeppelin 新注意.
5. 在Zeppelin Note在新备注中输入以下查询：

```
%flink.ssql(type=update)
select * from stock
```

   选择运行图标。

   此应用程序将显示来自 Amazon MSK 集群的数据。

要为应用程序打开 Apache Flink 控制面板以查看运营方面，请选择FLINK 工作. 有关 Flink 控制面板的更多信息，请参阅Apache Flink 控制面板中的Kinesis Data Analytics 开发者指南.

有关 Flink Stream SQL 查询的更多示例，请参阅查询中的Apache Flink 文档.

清理应用程序和依赖资源

删除 Studio 笔记本

1. 打开 Kinesis Data Analytics 控制台。
2. 选择MyNotebook.
3. 选择操作，那么Delete.

删除您的Amazon Glue数据库和连接

1. 打开Amazon Glue控制台https://console.aws.amazon.com/glue/.
2. 选择数据库从左侧导航栏中开始。选中旁边的复选框默认值来选择它。选择操作、删除数据库. 确认您的选择。
3. 选择连接从左侧导航栏中开始。选中旁边的复选框ZeppelinConnection来选择它。选择操作、删除连接. 确认您的选择。

删除您的 IAM 角色和策略

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择角色从左侧导航栏中开始。
3. 使用搜索栏搜索ZeppelinRole角色。
4. 选择ZeppelinRole角色。选择删除角色. 确认删除操作。

删除您的CloudWatch日志组

控制台创建CloudWatch在使用控制台创建应用程序时，将为您记录组和日志流。如果您使用Amazon CLI.

1. 打开 CloudWatch 控制台 (https://console.aws.amazon.com/cloudwatch/)。
2. 选择日志组从左侧导航栏中开始。
3. 选择/aws/kinesis-分析/MyNotebook日志组。
4. 依次选择 Actions（操作）和 Delete log group(s)（删除日志组）。确认删除操作。

清除 Kinesis Data Streams 资源

要删除 Kinesis 流，请打开 Kinesis Data Streams 控制台，选择您的 Kinesis 流，然后选择操作、Delete.

清除 MSK 资源

如果您为本教程创建 Amazon MSK 集群，请按照本部分中的步骤执行操作。本节介绍了清理 Amazon EC2 客户端实例、Amazon VPC 和 Amazon MSK 集群的说明。

删除您的亚马逊 MSK 集群

如果您为本教程创建了 Amazon MSK 集群，请按照以下步骤操作。

1. 在打开 Amazon MSK 控制台https://console.aws.amazon.com/msk/home?region=us-east-1#/home/.
2. 选择AmazonKafkaTutorial群集. 请选择 Delete（删除）。Enter**`delete`**在出现的窗口中，然后确认您的选择。

终止您的客户端实例

如果您为本教程创建了 Amazon EC2 客户端实例，请按照以下步骤操作。

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 选择实例从左侧导航栏中开始。
3. 选中旁边的复选框ZeppelinClient来选择它。
4. 选择实例状态、终止实例.

删除您的 Amazon VPC

如果您为本教程创建了 Amazon VPC，请按照以下步骤操作。

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 选择网络接口从左侧导航栏中开始。
3. 在搜索栏中输入您的 VPC ID，然后按 Enter 搜索。
4. 选中表标题中的复选框以选择所有显示的网络接口。
5. 依次选择操作、分离。在显示的窗口中，选择启用下强制支队. 选择Detach，等待所有网络接口到达Available状态。
6. 选中表标题中的复选框可再次选择所有显示的网络接口。
7. 依次选择 Actions（操作）和 Delete（删除）。确认该操作。
8. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
9. SelectAmazonKafkaTutorialVPC. 选择操作、删除 VPC. Enter**`delete`**并确认删除操作。

教程：部署为具有持久状态的应用程序

以下教程演示了如何将 Studio 笔记本部署为具有持久状态的 Kinesis Data Analytics 应用程序。

本教程包含以下部分：

- 设置 (p. 61)
- 使用Amazon Web Services Management Console (p. 61)
- 使用 “持久” 状态部署应用程序Amazon CLI (p. 62)

设置

按照创建 Studio 笔记本教程 (p. 48)，使用 Kinesis Data Streams 或亚马逊 MSK。为 Studio 笔记本命名`ExampleTestDeploy`.

使用Amazon Web Services Management Console

1. 在 S3 存储桶中添加您希望将打包代码存储在下的 S3 存储桶位置应用程序代码位置-可选的在控制台中。这使得这些步骤可以直接从笔记本电脑部署和运行应用程序。
2. 向应用程序角色添加所需的权限，以启用用于读取和写入 Amazon S3 存储桶的角色以及启动 Kinesis Data Analytics 应用程序：
   - 亚马逊 S3FullAccess
   - AmazonKinesisAnalyticsFull访问
   - 访问源、目的地和 VPC（如适用）。有关更多信息，请参阅 Studio Notebooks I (p. 44)。
3. 使用下面的示例代码：

```
%flink.ssql(type=update)
CREATE TABLE exampleoutput (
  'ticket' VARCHAR,
  'price' DOUBLE
```

```
)
WITH (
  'connector' = 'kinesis',
  'stream' = 'ExampleOutputStream',
  'aws.region' = 'us-east-1',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'json'
);

INSERT INTO exampleoutput SELECT ticker, price FROM exampleinputstream
```

4. 启动此功能后，您将在笔记本电脑中每个笔记的右上角看到一个新的下拉菜单，其中包含笔记本的名称。您可执行以下操作：

   - 查看 Studio 笔记本设置Amazon Web Services Management Console.
   - 构建您的齐柏林笔记并将其导出到 Amazon S3。此时，为你的应用程序提供一个名称然后选择构建和导出. 导出完成后，您将收到通知。
   - 如果需要，您可以在 Amazon S3 中查看并对可执行文件运行任何其他测试。
   - 构建完成后，您将能够将代码部署为具有持久状态和自动扩展的 Kinesis 流媒体应用程序。
   - 使用下拉菜单并选择将齐柏林注释部署为 Kinesis 流媒体应用程序. 查看应用程序名称并选择通过部署Amazon控制台.
   - 这将导致你进入Amazon Web Services Management Console用于创建 Kinesis Data Analytics 应用程序的页面。请注意，应用程序名称、并行性、代码位置、默认 Glue DB、VPC（如果适用）和 IAM 角色已预填充。验证 IAM 角色是否具有对源和目标的所需权限。默认情况下，为持久的应用程序状态管理启用快照
   - 选择创建应用.
   - 您可以选择配置然后修改任何设置，然后选择运行启动您的流式处理应用程序。

使用 “持久” 状态部署应用程序Amazon CLI

使用Amazon CLI，您必须更新Amazon CLI以使用随 Beta 2 信息提供的服务模型。有关如何使用更新后的服务模型的信息，请参阅设置 (p. 49).

以下示例代码创建了一个新的 Studio 笔记本：

```
aws kinesisanalyticsv2 create-application \
     --application-name <app-name> \
     --runtime-environment ZEPPELIN-FLINK-2_0 \
     --application-mode INTERACTIVE \
     --service-execution-role <iam-role>
     --application-configuration '{
       "ZeppelinApplicationConfiguration": {
         "CatalogConfiguration": {
           "GlueDataCatalogConfiguration": {
             "DatabaseARN": "arn:aws:glue:us-east-1:<account>:database/<glue-database-
name>"
           }
         }
       },
       "FlinkApplicationConfiguration": {
         "ParallelismConfiguration": {
           "ConfigurationType": "CUSTOM",
           "Parallelism": 4,
           "ParallelismPerKPU": 4
         }
       },
       "DeployAsApplicationConfiguration": {
            "S3ContentLocation": {
               "BucketARN": "arn:aws:s3:::<s3bucket>",
```

```
                "BasePath": "/something/"
            }
        },
       "VpcConfigurations": [
         {
           "SecurityGroupIds": [
             "<security-group>"
           ],
           "SubnetIds": [
             "<subnet-1>",
             "<subnet-2>"
           ]
         }
       ]
     }' \
     --region us-east-1
```

以下代码示例启动 Studio 笔记本：

```
aws kinesisanalyticsv2 start-application \
    --application-name <app-name> \
    --region us-east-1 \
    --no-verify-ssl
```

以下代码返回应用程序的 Apache Zeppelin 笔记本页面的 URL：

```
aws kinesisanalyticsv2 create-application-presigned-url \
    --application-name <app-name> \
    --url-type ZEPPELIN_UI_URL \

    --region us-east-1 \
    --no-verify-ssl
```

示例

以下示例查询演示如何在 Studio 笔记本中使用窗口查询来分析数据。

有关 Apache Link SQL 查询设置的信息，请参阅Flink 在齐柏林飞艇笔记本上进行交互式数据分析.

要在 Apache Flink 控制面板中查看您的应用程序，请选择FLINK JOB在应用程序中Zeppelin页.

有关窗口查询的更多信息，请参阅Windows中的Apache Flink 文档.

有关 Amaze Link 流式传输查询的更多示例，请参阅查询中的Apache Flink 文档.

使用亚马逊 MSK/Apache Kafka 创建表

您可以将 Amazon MSK Flink 连接器与 Kinesis Data Analytics Studio 配合使用，通过纯文本、SSL 或 IAM 身份验证对连接进行身份验证。根据您的要求使用特定属性创建表。

```
-- Plaintext connection

CREATE TABLE your_table (
  `column1` STRING,
  `column2` BIGINT
) WITH (
  'connector' = 'kafka',
  'topic' = 'your_topic',
  'properties.bootstrap.servers' = '<bootstrap servers>',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);

-- SSL connection

CREATE TABLE your_table (
  `column1` STRING,
  `column2` BIGINT
) WITH (
  'connector' = 'kafka',
   'topic' = 'your_topic',
  'properties.bootstrap.servers' = '<bootstrap servers>',
  'properties.security.protocol' = 'SSL',
  'properties.ssl.truststore.location' = '/usr/lib/jvm/java-11-amazon-corretto/lib/
security/cacerts',
  'properties.ssl.truststore.password' = 'changeit',
  'properties.group.id' = 'myGroup',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);

-- IAM connection (or for MSK Serverless)

CREATE TABLE your_table (
  `column1` STRING,
  `column2` BIGINT
) WITH (
  'connector' = 'kafka',
  'topic' = 'your_topic',
  'properties.bootstrap.servers' = '<bootstrap servers>',
  'properties.security.protocol' = 'SASL_SSL',
  'properties.sasl.mechanism' = 'AWS_MSK_IAM',
  'properties.sasl.jaas.config' = 'software.amazon.msk.auth.iam.IAMLoginModule required;',
  'properties.sasl.client.callback.handler.class' =
 'software.amazon.msk.auth.iam.IAMClientCallbackHandler',
  'properties.group.id' = 'myGroup',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);
```

您可以将这些属性与其他属性组合使用Apache Kafka SQL.

使用 Kinesis 创建表

在以下示例中，您可以使用 Kinesis Data 表：

```
CREATE TABLE KinesisTable (
```

```
  `column1` BIGINT,
  `column2` BIGINT,
  `column3` BIGINT,
  `column4` STRING,
  `ts` TIMESTAMP(3)
)
PARTITIONED BY (column1, column2)
WITH (
  'connector' = 'kinesis',
  'stream' = 'test_stream',
  'aws.region' = '<region>',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'csv'
);
```

有关您可以使用的其他属性的更多信息，请参阅。Amazon Kinesis Data Streams.

滚动窗口

下面的 Flink Streaming SQL 查询从ZeppelinTopic表:

```
%flink.ssql(type=update)
SELECT TUMBLE_END(event_time, INTERVAL '5' SECOND) as winend, MAX(price) as
 five_second_high, ticker
FROM ZeppelinTopic
GROUP BY ticker, TUMBLE(event_time, INTERVAL '5' SECOND)
```

滑动窗口

以下 Apache Flink Streaming SQL 查询从ZeppelinTopic表:

```
%flink.ssql(type=update)
SELECT HOP_END(event_time, INTERVAL '3' SECOND, INTERVAL '5' SECOND) AS winend, MAX(price)
 AS sliding_five_second_max
FROM ZeppelinTopic//or your table name in Amazon Glue
GROUP BY HOP(event_time, INTERVAL '3' SECOND, INTERVAL '5' SECOND)
```

交互式 SL

此示例打印事件时间和处理时间的最大值以及键值表中值的总和。确保您有来自the section called “数据生成器” (p. 66)运行。要尝试其他 SQL 查询，例如在 Studio 笔记本中进行筛选和联接，请参阅 Apache Flink 文档：查询在 Apache Flink 文档中。

```
%flink.ssql(type=single, parallelism=4, refreshInterval=1000, template=<h1>{2}</h1> records
 seen until <h1>Processing Time: {1}</h1> and <h1>Event Time: {0}</h1>)

-- An interactive query prints how many records from the `key-value-stream` we have seen so
 far, along with the current processing and event time.
SELECT
  MAX(`et`) as `et`,
  MAX(`pt`) as `pt`,
  SUM(`value`) as `sum`
FROM
  `key-values`
```

```
%flink.ssql(type=update, parallelism=4, refreshInterval=1000)

-- An interactive tumbling window query that displays the number of records observed per
 (event time) second.
```

```
-- Browse through the chart views to see different visualizations of the streaming result.
SELECT
  TUMBLE_START(`et`, INTERVAL '1' SECONDS) as `window`,
  `key`,
  SUM(`value`) as `sum`
FROM
  `key-values`
GROUP BY
  TUMBLE(`et`, INTERVAL '1' SECONDS),
  `key`;
```

BlackHole Sala 连接器

这些区域有： BlackHole SQL 连接器不需要您创建 Kinesis 数据流或 Amazon MSK 集群来测试查询。有关的信息 BlackHoleSQL 连接器，请参见BlackHole Sala 连接器在 Apache Flink 文档中。在此示例中，默认目录是内存中的目录。

```
%flink.ssql

CREATE TABLE default_catalog.default_database.blackhole_table (
 `key` BIGINT,
 `value` BIGINT,
 `et` TIMESTAMP(3)
) WITH (
 'connector' = 'blackhole'
)
```

```
%flink.ssql(parallelism=1)

INSERT INTO `test-target`
SELECT
  `key`,
  `value`,
  `et`
FROM
  `test-source`
WHERE
  `key` > 3
```

```
%flink.ssql(parallelism=2)

INSERT INTO `default_catalog`.`default_database`.`blackhole_table`
SELECT
  `key`,
  `value`,
  `et`
FROM
  `test-target`
WHERE
  `key` > 7
```

数据生成器

该示例使用 Scala 生成示例数据。您可以使用此示例数据测试各种查询。使用 create table 语句创建键值表。

```
import org.apache.flink.streaming.api.functions.source.datagen.DataGeneratorSource
import org.apache.flink.streaming.api.functions.source.datagen.RandomGenerator
import org.apache.flink.streaming.api.scala.DataStream
```

```
import java.sql.Timestamp

// ad-hoc convenience methods to be defined on Table
implicit class TableOps[T](table: DataStream[T]) {
    def asView(name: String): DataStream[T] = {
      if (stenv.listTemporaryViews.contains(name)) {
        stenv.dropTemporaryView("`" + name + "`")
      }
      stenv.createTemporaryView("`" + name + "`", table)
      return table;
    }
}
```

```
%flink(parallelism=4)
val stream = senv
 .addSource(new DataGeneratorSource(RandomGenerator.intGenerator(1, 10), 1000))
 .map(key => (key, 1, new Timestamp(System.currentTimeMillis)))
 .asView("key-values-data-generator")
```

```
%flink.ssql(parallelism=4)
-- no need to define the paragraph type with explicit parallelism (such as
 "%flink.ssql(parallelism=2)")
-- in this case the INSERT query will inherit the parallelism of the of the above paragraph
INSERT INTO `key-values`
SELECT
 `_1` as `key`,
 `_2` as `value`,
 `_3` as `et`
FROM
 `key-values-data-generator`
```

交互式Sala

这是 Scala 对the section called “交互式 SL” (p. 65). 有关更多 Scala 示例的信息，请参阅。表 API在 Apache Flink 文档中。

```
%flink
import org.apache.flink.api.scala._
import org.apache.flink.table.api._
import org.apache.flink.table.api.bridge.scala._

// ad-hoc convenience methods to be defined on Table
implicit class TableOps(table: Table) {
    def asView(name: String): Table = {
      if (stenv.listTemporaryViews.contains(name)) {
        stenv.dropTemporaryView(name)
      }
      stenv.createTemporaryView(name, table)
      return table;
    }
}
```

```
%flink(parallelism=4)

// A view that computes many records from the `key-values` we have seen so far, along with
 the current processing and event time.
val query01 = stenv
  .from("`key-values`")
  .select(
```

```
    $"et".max().as("et"),
    $"pt".max().as("pt"),
    $"value".sum().as("sum")
  ).asView("query01")
```

```
%flink.ssql(type=single, parallelism=16, refreshInterval=1000, template=<h1>{2}</h1>
 records seen until <h1>Processing Time: {1}</h1> and <h1>Event Time: {0}</h1>)

-- An interactive query prints the query01 output.
SELECT * FROM query01
```

```
%flink(parallelism=4)

// An tumbling window view that displays the number of records observed per (event time)
 second.
val query02 = stenv
  .from("`key-values`")
  .window(Tumble over 1.seconds on $"et" as $"w")
  .groupBy($"w", $"key")
  .select(
    $"w".start.as("window"),
    $"key",
    $"value".sum().as("sum")
  ).asView("query02")
```

```
%flink.ssql(type=update, parallelism=4, refreshInterval=1000)

-- An interactive query prints the query02 output.
-- Browse through the chart views to see different visualizations of the streaming result.
SELECT * FROM `query02`
```

交互式Par

这是 Python 对the section called “交互式 SL” (p. 65). 有关更多 Python 示例，请参阅。表 API在 Apache Flink 文档中。

```
%flink.pyflink
from pyflink.table.table import Table

def as_view(table, name):
  if (name in st_env.list_temporary_views()):
    st_env.drop_temporary_view(name)
  st_env.create_temporary_view(name, table)
  return table

Table.as_view = as_view
```

```
%flink.pyflink(parallelism=16)

# A view that computes many records from the `key-values` we have seen so far, along with
 the current processing and event time
st_env \
  .from_path("`keyvalues`") \
  .select(", ".join([
    "max(et) as et",
    "max(pt) as pt",
    "sum(value) as sum"
  ])) \
  .as_view("query01")
```

```
%flink.ssql(type=single, parallelism=16, refreshInterval=1000, template=<h1>{2}</h1>
 records seen until <h1>Processing Time: {1}</h1> and <h1>Event Time: {0}</h1>)

-- An interactive query prints the query01 output.
SELECT * FROM query01
```

```
%flink.pyflink(parallelism=16)

# A view that computes many records from the `key-values` we have seen so far, along with
 the current processing and event time
st_env \
  .from_path("`key-values`") \
  .window(Tumble.over("1.seconds").on("et").alias("w")) \
  .group_by("w, key") \
  .select(", ".join([
    "w.start as window",
    "key",
    "sum(value) as sum"
  ])) \
  .as_view("query02")
```

```
%flink.ssql(type=update, parallelism=16, refreshInterval=1000)

-- An interactive query prints the query02 output.
-- Browse through the chart views to see different visualizations of the streaming result.
SELECT * FROM `query02`
```

交互式 Python、SQL 和 Scala

您可以在笔记本中使用 SQL、Python 和 Scala 的任意组合进行交互式分析。在计划部署为具有持久状态的应用程序的 Studio 笔记本中，您可以使用 SQL 和 Scala 的组合。此示例显示了被忽略的部分以及在具有持久状态的应用程序中部署的部分。

```
%flink.ssql
CREATE TABLE `default_catalog`.`default_database`.`my-test-source` (
  `key` BIGINT NOT NULL,
  `value` BIGINT NOT NULL,
  `et` TIMESTAMP(3) NOT NULL,
  `pt` AS PROCTIME(),
  WATERMARK FOR `et` AS `et` - INTERVAL '5' SECOND
)
WITH (
  'connector' = 'kinesis',
  'stream' = 'kda-notebook-example-test-source-stream',
  'aws.region' = 'eu-west-1',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'json',
  'json.timestamp-format.standard' = 'ISO-8601'
)
```

```
%flink.ssql
CREATE TABLE `default_catalog`.`default_database`.`my-test-target` (
  `key` BIGINT NOT NULL,
  `value` BIGINT NOT NULL,
  `et` TIMESTAMP(3) NOT NULL,
  `pt` AS PROCTIME(),
  WATERMARK FOR `et` AS `et` - INTERVAL '5' SECOND
)
WITH (
```

```
  'connector' = 'kinesis',
  'stream' = 'kda-notebook-example-test-target-stream',
  'aws.region' = 'eu-west-1',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'json',
  'json.timestamp-format.standard' = 'ISO-8601'
)
```

```
%flink()

// ad-hoc convenience methods to be defined on Table
implicit class TableOps(table: Table) {
  def asView(name: String): Table = {
    if (stenv.listTemporaryViews.contains(name)) {
      stenv.dropTemporaryView(name)
    }
    stenv.createTemporaryView(name, table)
    return table;
  }
}
```

```
%flink(parallelism=1)
val table = stenv
  .from("`default_catalog`.`default_database`.`my-test-source`")
  .select($"key", $"value", $"et")
  .filter($"key" > 10)
  .asView("query01")
```

```
%flink.ssql(parallelism=1)

-- forward data
INSERT INTO `default_catalog`.`default_database`.`my-test-target`
SELECT * FROM `query01`
```

```
%flink.ssql(type=update, parallelism=1, refreshInterval=1000)

-- forward data to local stream (ignored when deployed as application)
SELECT * FROM `query01`
```

```
%flink

// tell me the meaning of life (ignored when deployed as application!)
print("42!")
```

跨账户 Kinesis 数据流

要使用除拥有 Studio 笔记本的账户之外的账户中的 Kinesis 数据流，请在运行 Studio 笔记本的账户中创建服务执行角色，并在具有数据流的账户中创建角色信任策略。使用`aws.credentials.provider`、`aws.credentials.role.arn`，和`aws.credentials.role.sessionName`在 Kinesis 连接器中的创建表 DDL 语句中，根据数据流创建表。

对 Studio 笔记本帐户使用以下服务执行角色。

```
{
 "Sid": "AllowNotebookToAssumeRole",
 "Effect": "Allow",
 "Action": "sts:AssumeRole"
```

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

使用AmazonKinesisFullAccess策略和数据流账户的以下角色信任策略。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::<accountID>:root"
 },
 "Action": "sts:AssumeRole",
 "Condition": {}
 }
 ]
}
```

对于 create table 语句，请使用以下段落。

```
%flink.ssql
CREATE TABLE test1 (
name VARCHAR,
age BIGINT
) WITH (
'connector' = 'kinesis',
'stream' = 'stream-assume-role-test',
'aws.region' = 'us-east-1',
'aws.credentials.provider' = 'ASSUME_ROLE',
'aws.credentials.role.arn' = 'arn:aws:iam::<accountID>:role/stream-assume-role-test-role',
'aws.credentials.role.sessionName' = 'stream-assume-role-test-session',
'scan.stream.initpos' = 'TRIM_HORIZON',
'format' = 'json'
)
```

比较适用于 SQL 应用程序的 Studio 笔记本和 Kinesis Data Analytics

Kinesis Data Analytics 提供 SQL 支持Kinesis Data Analytics. 以下功能适用于 Studio 笔记本电脑，但不适用于适用于 SQL 应用程序的 Kinesis Data Analytics：

- 在多个 Kinesis 数据流之间或在 Kinesis 数据流与 Amazon MSK 主题之间联接流数据
- 数据流中已转换数据的实时可视化
- 在同一个应用程序中使用 Python 脚本或 Scala 程序

问题排查

本节包含 Studio 笔记本电脑的故障排除信息。

停止卡住的应用程序

要停止处于暂时状态的应用程序，请调用StopApplication动作Forceparameter settrue. 有关更多信息，请参阅 。运行应用程序中的Kinesis Data Analytics.

取消任务

本节将向您展示如何取消无法从 Apache Zeppelin 获得的 Apache Flink 作业。如果你想取消这样的作业，请转到 Apache Flink 控制面板，复制作业 ID，然后在以下示例中使用它。

取消单个作业：

```
%flink.pyflink
import requests

requests.patch("https://zeppelin-flink:8082/jobs/[job_id]", verify=False)
```

要取消所有运行的作业，请执行以下操作：

```
%flink.pyflink
import requests

r = requests.get("https://zeppelin-flink:8082/jobs", verify=False)
jobs = r.json()['jobs']

for job in jobs:
    if (job["status"] == "RUNNING"):
        print(requests.patch("https://zeppelin-flink:8082/jobs/{}".format(job["id"]),
 verify=False))
```

要取消所有作业，请执行以下操作：

```
%flink.pyflink
import requests

r = requests.get("https://zeppelin-flink:8082/jobs", verify=False)
jobs = r.json()['jobs']

for job in jobs:
    requests.patch("https://zeppelin-flink:8082/jobs/{}".format(job["id"]), verify=False)
```

重新启动 Apache Flink 解释器

在你的 Studio 笔记本中重新启动 Apache Flink 解释器

1. 选择配置在屏幕的右上角附近。
2. 选择解释.
3. 选择重新启动然后确定。.

附录：创建自定义 IAM 策略

您通常使用托管 IAM 策略来允许应用程序访问依赖资源。如果您需要对应用程序的权限进行更精细的控制，可以使用自定义 IAM 策略。本部分包含自定义 IAM 策略示例。

> **Note**
>
> 在以下策略示例中，将占位符文本替换为应用程序的值。

本主题包含下列部分：

Amazon Glue

以下示例策略授予权限以访问Amazon Glue数据库。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GlueTable",
            "Effect": "Allow",
            "Action": [
                "glue:GetConnection",
                "glue:GetTable",
                "glue:GetTables",
                "glue:GetDatabase",
                "glue:CreateTable",
                "glue:UpdateTable"
            ],
            "Resource": [
                "arn:aws:glue:<region>:<accountId>:connection/*",
                "arn:aws:glue:<region>:<accountId>:table/<database-name>/*",
                "arn:aws:glue:<region>:<accountId>:database/<database-name>",
                "arn:aws:glue:<region>:<accountId>:database/hive",
                "arn:aws:glue:<region>:<accountId>:catalog"
            ]
        },
        {
            "Sid": "GlueDatabase",
            "Effect": "Allow",
            "Action": "glue:GetDatabases",
            "Resource": "*"
        }
    ]
}
```

CloudWatch 日志

以下策略将授予访问的权限 CloudWatch 日志：

```
{
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:<region>:<accountId>:log-group:*"
      ]
    },
    {
      "Sid": "ListCloudwatchLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
```

```
      "Resource": [
        "<logGroupArn>:log-stream:*"
      ]
    },
    {
      "Sid": "PutCloudwatchLogs",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "<logStreamArn>"
      ]
    }
```

Note

如果您使用控制台创建应用程序，则控制台会添加必要的策略来访问 CloudWatch 将日志记录到您的应用程序角色。

Kinesis Streams

您的应用程序可以将 Kinesis 流用于源或目标。您的应用程序需要读取权限才能从源流进行读取，而写入权限才能写入目标流。

以下策略授予从用作源的 Kinesis 流进行读取的权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "KinesisShardDiscovery",
      "Effect": "Allow",
      "Action": "kinesis:ListShards",
      "Resource": "*"
    },
    {
      "Sid": "KinesisShardConsumption",
      "Effect": "Allow",
      "Action": [
        "kinesis:GetShardIterator",
        "kinesis:GetRecords",
        "kinesis:DescribeStream",
        "kinesis:DescribeStreamSummary",
        "kinesis:RegisterStreamConsumer",
        "kinesis:DeregisterStreamConsumer"
      ],
      "Resource": "arn:aws:kinesis:<region>:<accountId>:stream/<stream-name>"
    },
    {
      "Sid": "KinesisEfoConsumer",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStreamConsumer",
        "kinesis:SubscribeToShard"
      ],
      "Resource": "arn:aws:kinesis:<region>:<account>:stream/<stream-name>/consumer/*"
    }
  ]
}
```

以下策略授予对用作目标的 Kinesis 流进行写入的权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Sid": "KinesisStreamSink",
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecord",
                "kinesis:PutRecords",
                "kinesis:DescribeStreamSummary",
                "kinesis:DescribeStream"
            ],
            "Resource": "arn:aws:kinesis:<region>:<accountId>:stream/<stream-name>"
        }
    ]
}
```

如果您的应用程序访问加密的 Kinesis 流，则必须授予其他权限才能访问该流和流的加密密钥。

以下策略授予访问加密源流和流加密密钥的权限：

```
{
      "Sid": "ReadEncryptedKinesisStreamSource",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "<inputStreamKeyArn>"
      ]
    }
    ,
```

以下策略授予访问加密目标流和流加密密钥的权限：

```
{
      "Sid": "WriteEncryptedKinesisStreamSink",
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "<outputStreamKeyArn>"
      ]
    }
```

Amazon MSK 集群

要授予对 Amazon MSK 集群的访问权限，您需要授予对该集群的 VPC 的访问权限。有关访问 Amazon VPC 的策略示例，请参阅VPC 应用程序权限.

使用 Amazon Kinesis Data Analytics Apache Flink 入门 (DataStreamAPI)

本节介绍针对 Apache Flink 的 Kinesis Data Analytics 的基本概念和DataStreamAPI。它介绍了可用于创建和测试应用程序的选项。它还提供了相应的说明以安装所需的工具，以完成本指南中的教程和创建第一个应用程序。

主题

Flink 应用程序的 Kinesis Data Analytics 的组件

为了处理数据，Kinesis Data Analytics 应用程序使用 Java/Apache Maven 或 Scala 应用程序，该应用程序使用 Apache Flink 运行时处理输入和生成输出。

Kinesis Data Analytics 应用程序具有以下组件：

- 运行时属性：您可以使用运行时属性配置应用程序，而无需重新编译应用程序代码。
- 源：应用程序通过使用资源. 源连接器从 Kinesis Data Streams、Amazon S3 存储桶等读取数据。有关更多信息，请参阅 源 (p. 8)。
- 运算符：应用程序使用一个或多个处理数据。运营商. 操作符可以转换、丰富或聚合数据。有关更多信息，请参阅 DataStreamAPI 操作符 (p. 12)。
- 接收器: 使用该应用程序将数据生成到外部源中的水槽. 接收器连接器将数据写入 Kinesis Data Streams、Amazon S3 存储桶等。有关更多信息，请参阅 接收器 (p. 9)。

在创建、编译和打包应用程序代码后，您可以将代码包上传到 Amazon Simple Storage Service (Amazon S3) 存储桶。然后，您创建 Kinesis Data Analytics 应用程序。您在代码包位置中传入 Kinesis 数据流以作为流数据源，它通常是接收应用程序处理的数据的流或文件位置。

完成练习的先决条件

要完成本指南中的步骤，您必须满足以下条件：

- Java 开发工具包 (JDK) 版本 11. 设置 `JAVA_HOME` 环境变量，使其指向您的 JDK 安装位置。
- 我们建议您使用开发环境（如 Eclipse Java Neon 或 IntelliJ Idea）来开发和编译您的应用程序。
- Git 客户端. 如果尚未安装 Git 客户端，请安装它。
- Apache Maven 编译器插件。Maven 必须位于您的有效路径中。要测试您的 Apache Maven 安装，请输入以下内容：

```
$ mvn -version
```

要开始，请转到第 1 步：设置Amazon创建管理员用户 (p. 77)。

第 1 步：设置Amazon创建管理员用户

首次使用 Kinesis Data Analytics 前，请完成以下任务：

1. 注册 Amazon (p. 77)
2. 创建 IAM 用户 (p. 77)

注册 Amazon

当你注册时Amazon时，您的账户会自动注册所有 Amazon 服务，包括 Kinesis Data Analytics 服务。您只需为使用的服务付费。

使用 Kinesis Data Analytics 时，您仅需为实际使用的资源付费。如果您是新手Amazon客户，您可以免费试用 Kinesis Data Analytics。有关更多信息，请参阅 Amazon 免费套餐。

如果您已有 Amazon 账户，请跳到下一个任务。如果没有Amazon账户，请执行以下步骤来创建一个。

创建 Amazon 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。

记住您的账户 ID，因为进行下一个任务时需要用到。

创建 IAM 用户

中的服务Amazon（如 Kinesis Data Analytics）要求您在访问时提供凭证。这样，服务才能确定您是否有权访问该服务所拥有的资源。Amazon Web Services Management Console要求您输入密码。

您可以为您的Amazon账户以访问Amazon Command Line Interface(Amazon CLI) 或者 API。但是，我们不建议使用您的Amazon账户的凭证访问Amazon。相反，我们建议您使用 Amazon Identity and Access Management (IAM)。创建 IAM 用户，将该用户添加到具有管理权限的 IAM 组，然后向您创建的 IAM 用户授予管理权限。您随后便可以使用一个特殊的 URL 和该 IAM 用户的凭证访问 Amazon。

如果你注册了Amazon，但您尚未为自己创建 IAM 用户，可以使用 IAM 控制台创建。

本指南中的入门练习假定您拥有具有管理员权限的用户 (`adminuser`)。请按照以下过程在您的账户中创建 `adminuser`。

为管理员创建组

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Groups (组)，然后选择 Create New Group (创建新组)。

3. 对于组名，输入组的名称，例如 **Administrators**，然后选择 下一步。
4. 在策略列表中，选中AdministratorAccess政策。您可以使用 Filter (筛选) 菜单和 Search (搜索) 框来筛选策略列表。
5. 选择 Next Step (下一步)，然后选择 Create Group (创建组)。

您的新组列在 Group Name 下方。

要为您自己创建 IAM 用户，请将用户添加到 Administrator 组中，然后创建密码

1. 在导航窗格中，选择 Users，然后选择 Add user。
2. 在 User name(用户名) 框中，输入一个用户名。
3. 选择这两者以编程方式访问和Amazon访问管理控制台.
4. 选择 Next:。Permissions (下一步：权限)。
5. 选中 Administrators 组旁的复选框。接下来，选择 Next (下一步)：审核。
6. 选择 Create user。

以新 IAM 用户身份登录

1. 注销 Amazon Web Services Management Console。
2. 使用下面的 URL 格式登录控制台：

   `https://aws_account_number.signin.aws.amazon.com/console/`

   这些区域有：*aws_account_number*是您的账户 ID 没有连字符。例如，如果您的账户 ID 是 1234-5678-9012，请替换*aws_account_number*和**123456789012**. 有关如何查找您的账号的更多信息，请参阅您的Amazon账户 ID 及其别名中的IAM 用户指南.
3. 输入您刚创建的 IAM 用户名和密码。登录后，导航栏将显示 *your_user_name* @ *your_aws_account_id*。

   Note

   如果您不希望您的登录页面 URL 包含 账户 ID，可以创建账户别名。

创建或删除账户别名

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格上，选择 Dashboard。
3. 查找 IAM 用户登录链接。
4. 要创建别名，请选择 Customize (自定义)。输入要用于别名的名称，然后选择 Yes, Create (是，创建)。
5. 要删除别名，请选择 Customize，然后选择 Yes, Delete。登录 URL 会恢复使用 账户 ID。

要在创建账户别名后登录，请使用以下 URL：

`https://your_account_alias.signin.aws.amazon.com/console/`

要为您的账户验证 IAM 用户的登录链接，请打开 IAM 控制台并在控制面板的 IAM 用户登录链接下进行检查。

有关 IAM 的更多信息，请参阅以下文档：

- Amazon Identity and Access Management (IAM)
- 入门

- IAM 用户指南

下一个步骤

第 2 步：设置 Amazon Command Line Interface (Amazon CLI) (p. 79)

第 2 步：设置 Amazon Command Line Interface (Amazon CLI)

在此步骤中，您将下载并配置Amazon CLI与 Kinesis Data Analytics 一起使用。

Note

本指南中的入门练习假定您使用账户中的管理员凭证 (`adminuser`) 来执行这些操作。

Note

如果您已安装 Amazon CLI，您可能需要升级以获得最新的功能。有关更多信息，请参阅 。安装 Amazon Command Line Interface中的Amazon Command Line Interface用户指南. 要检查 Amazon CLI 的版本，请运行以下命令：

```
aws --version
```

本教程中的练习需要以下 Amazon CLI 版本或更高版本：

```
aws-cli/1.16.63
```

设置 Amazon CLI

1. 下载并配置 Amazon CLI。有关说明，请参阅《Amazon Command Line Interface 用户指南》中的以下主题：

   - 安装 Amazon Command Line Interface
   - 配置 Amazon CLI

2. 在 Amazon CLI `config` 文件中为管理员用户添加一个命名的配置文件。在执行 Amazon CLI 命令时，您将使用此配置文件。有关指定配置文件的更多信息，请参命名配置文件中的Amazon Command Line Interface用户指南.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

   有关可用列表Amazon地区，请参阅区域和终端节点中的Amazon Web Services 一般参考.

   Note

   本教程中的示例代码和命令使用美国西部（俄勒冈）区域。要使用不同的区域，请将本教程的代码和命令中的区域更改为要使用的区域。

3. 在命令提示符处输入以下帮助命令来验证设置：

   ```
   aws help
   ```

在你设置之后Amazon账户和Amazon CLI中，您可以尝试下一个练习中，在其中，您将配置一个示例应用程序并测试end-to-end设置。

下一个步骤

第 3 步：创建和运行 Apache Flink 应用程序的 Kinesis Data Analytics 应用程序 (p. 80)

第 3 步：创建和运行 Apache Flink 应用程序的 Kinesis Data Analytics 应用程序

在本练习中，您创建 Kinesis Data Analytics 应用程序，它将数据流作为源和接收器。

本节包含以下步骤：

创建两个 Amazon Kinesis Data Streams

在为本练习创建 Kinesis Data Analytics 应用程序前，请创建两个 Kinesis Data Analytics 应用程序（`ExampleInputStream`和`ExampleOutputStream`)。您的应用程序将这些数据流用于应用程序源和目标流。

您可以使用 Amazon Kinesis Console 或以下内容创建这些流：Amazon CLI命令。有关控制台说明，请参阅创建和更新数据流中的Amazon Kinesis Data Streams 开发人员指南.

创建数据流 (Amazon CLI)

1. 创建第一个直播 (`ExampleInputStream`)，请使用以下 Amazon Kinesis`create-stream` Amazon CLI命令。

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. 要创建应用程序用来写入输出的第二个流，请运行同一命令（将流名称更改为 `ExampleOutputStream`）。

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入流，以供应用程序处理。

Note

此部分需要 Amazon SDK for Python (Boto)。

1. 使用以下内容创建名为 `stock.py` 的文件：

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

2. 在本教程的后面部分，您运行 `stock.py` 脚本，以将数据发送到应用程序。

```
$ python stock.py
```

下载并检查 Apache Flink 流式处理 Java 代码

此示例的 Java 应用程序代码可从GitHub. 要下载应用程序代码，请执行以下操作：

1. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

2. 导航到 `amazon-kinesis-data-analytics-java-examples/GettingStarted`目录。

请注意有关应用程序代码的以下信息：

- 一个项目对象模型 (pom.xml)文件包含有关应用程序的配置和依赖项的信息，包括 Kinesis Data Analytics 库。
- `BasicStreamingJob.java` 文件包含定义应用程序功能的 `main` 方法。

- 该应用程序使用 Kinesis Source 从源流中进行读取。以下代码段创建 Kinesis 源：

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- 您的应用程序使用 `StreamExecutionEnvironment` 对象创建源和接收连接器以访问外部资源。
- 该应用程序将使用静态属性创建源和接收连接器。要使用动态应用程序属性，请使用 `createSourceFromApplicationProperties` 和 `createSinkFromApplicationProperties` 方法以创建连接器。这些方法读取应用程序的属性来配置连接器。

  有关运行时属性的更多信息，请参阅运行时属性 (p. 18)。

编译应用程序代码

在本节中，您使用 Apache Maven 编译器创建应用程序的 Java 代码。有关安装 Apache Maven 和 Java 开发工具包 (JDK) 的信息，请参阅完成练习的先决条件 (p. 76)。

编译应用程序代码

1. 要使用您的应用程序代码，您将其编译和打包成 JAR 文件。您可以通过两种方式之一编译和打包您的代码：

   - 使用命令行 Maven 工具。在包含 `pom.xml` 文件的目录中通过运行以下命令创建您的 JAR 文件：

     ```
     mvn package -Dflink.version=1.13.2
     ```

   - 设置开发环境。有关详细信息，请参阅您的开发环境文档。

     Note

     提供的源代码依赖于 Java 11 中的库。

   您可以作为 JAR 文件上传您的包，也可以将包压缩为 ZIP 文件并上传。如果您使用 Amazon CLI 创建应用程序，您可以指定您的代码内容类型（JAR 或 ZIP）。
2. 如果编译时出错，请验证 `JAVA_HOME` 环境变量设置正确。

如果应用程序成功编译，则创建以下文件：

`target/aws-kinesis-analytics-java-apps-1.0.jar`

上传 Apache Flink 流式处理 Java 代码

在本节中，您创建 Amazon Simple Storage Service (Amazon S3) 存储桶并上传应用程序代码。

上传应用程序代码

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. Enter**`ka-app-code-<username>`**中的Bucket name字段中返回的子位置类型。将后缀（如您的用户名）添加到存储桶名称，以使其具有全局唯一性。请选择 Next (下一步)。
4. 在配置选项步骤中，让设置保持原样，然后选择下一步。
5. 在设置权限步骤中，让设置保持原样，然后选择下一步。
6. 请选择 Create bucket (创建存储桶)。
7. 在 Amazon S3 控制台中，选择ka-app-code-***`<username>`***存储桶，然后选择上传.

8. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `aws-kinesis-analytics-java-apps-1.0.jar` 文件。请选择 Next (下一步)。
9. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在 Amazon S3 存储桶中，应用程序可以在其中访问代码。

创建和运行 Kinesis Data Analytics 应用程序

您可以使用控制台或创建和运行 Kinesis Data Analytics 应用程序。Amazon CLI.

> Note
>
> 当您使用控制台创建应用程序时，您的Amazon Identity and Access Management(IAM) 和亚马逊CloudWatch日志资源是为你创建的。当您使用 Amazon CLI 创建应用程序时，您可以单独创建这些资源。

主题

创建并运行应用程序（控制台）

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于描述，输入 **`My java test app`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。
   - 将版本下拉列表保留为Apache Flink 版本 1.13.2（推荐版本）.
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
5. 选择 Create application（创建应用程序）。

> Note
>
> 使用控制台创建 Kinesis Data Analytics 应用程序时，您可以选择为您的应用程序创建 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源使用您的应用程序名称和区域命名，如下所示：
>
> - 策略：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - 角色：`kinesis-analytics-`*`MyApplication`*`-`*`us-west-2`*

编辑 IAM 策略

编辑 IAM 策略以添加访问 Kinesis Data Streams 的权限。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。

2. 选择策略。选择控制台在上一部分中为您创建的 **kinesis-analytics-service-MyApplication-us-west-2** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。
4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
```

```
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**ka-app-code-*<username>***.
   - 适用于Amazon S3 对象的路径输入**aws-kinesis-analytics-java-apps-1.0.jar**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
4. 在 Properties (属性) 下，对于 Group ID (组 ID)，输入 **ProducerConfigProperties**。
5. 输入以下应用程序属性和值：

密钥	值
flink.inputstream.initpos	**LATEST**
aws.region	**us-west-2**
AggregationEnabled	**false**

6. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
7. 适用于CloudWatch记录在中，选择启用”复选框。
8. 选择 Update (更新) 。

Note

当你选择启用亚马逊CloudWatch记录时，Kinesis Data Analytics 会为您创建日志组和日志流。这些资源的名称如下所示：

- 日志组：`/aws/kinesis-analytics/MyApplication`
- 日志流：`kinesis-analytics-log-stream`

运行应用程序

可以通过运行应用程序、打开 Apache Flink 控制面板并选择所需的 Flink 作业来查看 Flink 作业图。

停止应用程序

在存储库的MyApplication页面上，选择停止. 确认该操作。

更新应用程序

使用控制台，您可以更新应用程序设置，例如应用程序属性、监控设置，或应用程序 JAR 文件的位置和文件名。如果您需要更新应用程序代码，还可以从 Amazon S3 存储桶重新加载应用程序 JAR。

在存储库的MyApplication页面上，选择配置. 更新应用程序设置，然后选择更新。

创建并运行应用程序 (Amazon CLI)

在本部分中，您将使用Amazon CLI创建和运行 Kinesis Data Analytics 应用程序。Apache Flink 的 Kinesis Data Analytics 使用`kinesisanalyticsv2` Amazon CLI命令来创建 Kinesis Data Analytics 应用程序并与之交互。

创建权限策略

> **Note**
>
> 您必须为应用程序创建一个权限策略和角色。如果未创建这些 IAM 资源，应用程序将无法访问其数据和日志流。

首先，使用两个语句创建权限策略：一个语句授予对源流执行 `read` 操作的权限，另一个语句授予对接收器流执行 `write` 操作的权限。然后，您将策略附加到 IAM 角色（在下一部分中将创建该角色）。因此，在 Kinesis Data Analytics 代入该角色时，服务具有必要的权限从源流进行读取和写入接收器流。

使用以下代码创建 `KAReadSourceStreamWriteSinkStream` 权限策略。Replace*username*使用您用于创建 Amazon S3 存储桶以存储应用程序代码的用户名。将 Amazon 资源名称 (ARN) 中的账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleOutputStream"
        }
    ]
}
```

适用于step-by-step有关创建权限策略的说明，请参阅教程：创建并附加您的第一个客户托管策略中的IAM 用户指南.

> **Note**
>
> 要访问其他亚马逊服务，您可以使用Amazon SDK for Java. Kinesis Data Analytics 会自动将开发工具包所需的凭证设置为与您的应用程序关联的服务执行 IAM 角色的凭证。无需执行其他步骤。

创建 IAM 角色

在本节中，您将创建一个 IAM 角色，Kinesis Data Analytics 应用程序可以代入此角色来读取源流和写入接收器流。

如果没有权限，Kinesis Data Analytics 无法访问您的流 您通过 IAM 角色授予这些权限。每个 IAM 角色附加了两个策略。此信任策略授予 Kinesis Data Analytics 权限代入该角色，权限策略确定 Kinesis Data Analytics 入该角色后可以执行的操作。

您将在上一部分中创建的权限策略附加到此角色。

创建 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles (角色) 和 Create Role (创建角色)。
3. UNDER选择受信任实体的类型，选择Amazon服务. 在 Choose the service that will use this role (选择将使用此角色的服务) 下，选择 Kinesis。在选择您的使用案例下，选择 Kinesis Analytics。

   选择 Next:。Permissions (下一步：权限)。
4. 在存储库的附加权限策略页面上，选择后续：审核。在创建角色后，您可以附加权限策略。
5. 在存储库的创建角色页面，输入**`KA-stream-rw-role`**(对于)Role name (角色名称). 请选择 Create role（创建角色）。

   现在，您已经创建了一个名为的新 IAM 角色`KA-stream-rw-role`. 接下来，您更新角色的信任和权限策略。
6. 将权限策略附加到角色。

   Note

   在本练习中，Kinesis Data Analytics 代入此角色，以便同时从 Kinesis 数据流（源）读取数据和将输出写入另一个 Kinesis 数据流。因此，您附加在上一步（the section called “创建权限策略” (p. 86)）中创建的策略。

   a. 在 Summary (摘要) 页上，选择 Permissions (权限) 选项卡。
   b. 选择附加策略。
   c. 在搜索框中，输入 **`KAReadSourceStreamWriteSinkStream`**（您在上一部分中创建的策略）。
   d. 选择KAReadSourceStreamWriteSinkStream策略，然后选择附加策略.

现在，您已经创建了应用程序用来访问资源的服务执行角色。记下新角色的 ARN。

适用于step-by-step有关创建角色的说明，请参阅创建 IAM 角色 (控制台)中的IAM 用户指南.

创建 Kinesis Data Analytics 应用程序

1. 将以下 JSON 代码保存到名为 `create_request.json` 的文件中。将示例角色 ARN 替换为您之前创建的角色的 ARN。将存储桶 ARN 后缀 (*`username`*) 替换为在前一部分中选择的后缀。将服务执行角色中的示例账户 ID (*`012345678901`*) 替换为您的账户 ID。

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_13",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/KA-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
```

```
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. 使用上述请求执行 `CreateApplication` 操作来创建应用程序：

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

应用程序现已创建。您在下一步中启动应用程序。

启动应用程序

在本节中，您使用 `StartApplication` 操作来启动应用程序。

启动应用程序

1. 将以下 JSON 代码保存到名为 `start_request.json` 的文件中。

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. 使用上述请求执行 `StartApplication` 操作来启动应用程序：

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

应用程序正在运行。您可以在亚马逊上查看 Kinesis Data Analytics 指标CloudWatch控制台，以验证应用程序是否正常工作。

停止应用程序

在本节中，您使用 `StopApplication` 操作来停止应用程序。

停止应用程序

1. 将以下 JSON 代码保存到名为 `stop_request.json` 的文件中。

```
{
   "ApplicationName": "test"
}
```

2. 使用下面的请求执行 StopApplication 操作来停止应用程序：

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

应用程序现已停止。

添加 CloudWatch 日志记录选项

您可以使用Amazon CLI添加亚马逊CloudWatch将流记录到应用程序中。有关使用的信息CloudWatch登录应用程序，请参阅the section called “设置日志记录” (p. 205).

更新环境属性

在本节中，您使用 UpdateApplication 操作更改应用程序的环境属性，而无需重新编译应用程序代码。在本示例中，您更改源流和目标流的区域。

更新应用程序的环境属性

1. 将以下 JSON 代码保存到名为 update_properties_request.json 的文件中。

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                     "flink.stream.initpos" : "LATEST",
                     "aws.region" : "us-west-2",
                     "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                     "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. 使用前面的请求执行 UpdateApplication 操作以更新环境属性：

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

更新应用程序代码

在您需要使用新版本的代码包更新应用程序代码时，您可以使用 UpdateApplication Amazon CLI 操作。

Note

要使用相同的文件名加载新版本的应用程序代码，您必须指定新的对象版本。有关使用 Amazon S3 对象版本的更多信息，请参阅启用或禁用版本控制.

使用Amazon CLI，请从 Amazon S3 存储桶中删除以前的代码包，上传新版本，然后调用`UpdateApplication`指定相同的 Amazon S3 存储桶和对象名称以及新的对象版本。应用程序将使用新的代码包重新启动。

以下示例 `UpdateApplication` 操作请求重新加载应用程序代码并重新启动应用程序。将 `CurrentApplicationVersionId` 更新为当前的应用程序版本。您可以使用 `ListApplications` 或 `DescribeApplication` 操作检查当前的应用程序版本。将存储桶名称后缀 (*<username>*) 更新为在the section called “创建两个 Amazon Kinesis Data Streams” (p. 80)一节中选择的后缀。

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

下一个步骤

第 4 步：清理 Amazon 资源 (p. 90)

第 4 步：清理 Amazon 资源

本节包含清理在入门教程中创建的 Amazon 资源的过程。

本主题包含下列部分：

- 删除 Kinesis Data Analytics 应用程序 (p. 90)
- 删除 Kinesis Data Streams (p. 91)
- 删除您的 Amazon S3 对象和存储桶 (p. 91)
- 删除 IAM 资源 (p. 91)
- 删除 CloudWatch 资源 (p. 91)
- 下一个步骤 (p. 91)

删除 Kinesis Data Analytics 应用程序

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 在应用程序的页面中，选择 Delete (删除)，然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Streams 面板中，选择ExampleInput流.
3. 在ExampleInput流页面上，选择删除 Kinesis Streams然后确认删除。
4. 在Kinesis Streams在页面上，选择ExampleOutput流，选择操作，选择Delete，然后确认删除。

删除您的 Amazon S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择ka-app-code-***<username>***存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除 IAM 资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-***<your-region>***政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏中，选择 Roles（角色）。
7. 选择kinesis-analyt#-MyApplication-***<your-region>***角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除 CloudWatch 资源

1. 打开 CloudWatch 控制台 (https://console.aws.amazon.com/cloudwatch/)。
2. 在导航栏中，选择 Logs (日志)。
3. 选择/aws/kinesis-analytic/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

下一个步骤

第 5 步：后续步骤 (p. 91)

第 5 步：后续步骤

现在您已创建并运行基本的 Kinesis Data Analytics 应用程序，请参阅以下资源了解更高级的 Kinesis Data Analytics 解决方案。

- 这些区域有：AmazonAmazon Kinesis 的流数据解决方案：这些区域有：Amazon适用于 Amazon Kinesis 的流数据解决方案自动配置Amazon轻松捕获、存储、处理和交付流数据所需的服务。该解决方案为解决流式数据使用案例提供了多种选择。Kinesis Data Analytics 选项提供了end-to-end直播 ETL 示例演示了对

模拟的纽约出租车数据运行分析操作的真实世界应用程序。该解决方案设置了所有必要Amazon资源，例如 IAM 角色和策略，CloudWatch以及控制面板CloudWatch警报。

- AmazonAmazon MSK 流式传输数据解决方案：这些区域有：Amazon亚马逊 MSK 流数据解决方案提供 Amazon CloudFormation数据流经生产者、流式存储、消费者和目的地的模板。
- 点击流实验室与 Apache Flink 和 Apache Kafka：一个端到端的实验室，该实验室用于点击流使用案例，使用适用于 Apache Kafka 的亚马逊托管流媒体进行流媒体存储和适用于 Apache Flink 的 Amazon Kinesis Data Analytics 应用程序用于流
- 直播分析研讨会：在本次研讨会中，你构建end-to-end用于近乎实时地采集、分析和可视化流式传输数据的流式架构。你开始改善纽约市一家出租车公司的运营。您可以近乎实时地分析纽约市出租车车队的遥测数据，以优化其车队运营。
- Kinesis Data Analytics 示例 (p. 123)：本开发人员指南的本节提供在 Kinesis Data Analytics 中创建和使用应用程序的示例。它们包括示例代码和step-by-step说明，以帮助您创建 Kinesis Data Analytics 应用程序和测试结果。
- 了解 Flink: 实施训练：官方入门 Apache Flink 培训，帮助您开始编写可扩展的流式 ETL、分析和事件驱动型应用程序。

  Note

  请注意，Kinesis Data Analytics 不支持本培训中使用的 Apache Flink 版本 (1.12)。你可以在 Flink Kinesis Data Analytics 中使用 Flink 1.13。

Amazon Kinesis Data Analytics for Apache Flink（表 API）入门

本节介绍 Kinesis Data Analytics for Apache Flink 和表 API 的基本概念。它介绍了可用于创建和测试应用程序的选项。它还提供了相应的说明以安装所需的工具，以完成本指南中的教程和创建第一个应用程序。

主题

Flink 应用程序的 Kinesis Data Analytics 的组件

为了处理数据，Kinesis Data Analytics 应用程序使用 Java/Apache Maven 或 Scala 应用程序，该应用程序使用 Apache Flink 运行时处理输入和生成输出。

Kinesis Data Analytics 应用程序具有以下组件：

- 运行时属性：您可以使用运行时属性配置应用程序，而无需重新编译应用程序代码。
- 表来源：应用程序通过源使用数据。一个资源连接器从 Kinesis 数据流、Amazon MSK 主题或类似主题读取数据。有关更多信息，请参阅 表 API 来源 (p. 13)。
- 函数：该应用程序使用一个或多个函数处理数据。一个功能可以转换、丰富或聚合数据。
- 接收器：应用程序使用接收器将生成的数据发送到外部源。一个下沉连接器将数据写入 Kinesis 数据流、Kinesis Data Firehose 传输流、Amazon MSK 主题、Amazon S3 存储桶等。有关更多信息，请参阅 表 API 接收器 (p. 14)。

在创建、编译和打包应用程序代码后，您将代码包上传到 Amazon S3 存储桶。然后创建 Kinesis Data Analytics 应用程序。您在代码包位置中传入一个 Amazon MSK 主题以作为流数据源，它通常是接收应用程序处理的数据的流或文件位置。

先决条件

在开始本教程之前，请完成使用 Amazon Kinesis Data Analytics Apache Flink 入门 (DataStreamAPI) (p. 76)：

- 第 1 步：设置Amazon创建管理员用户 (p. 77)
- 第 2 步：设置 Amazon Command Line Interface (Amazon CLI) (p. 79)

要开始使用，请参阅创建应用程序 (p. 94).

为 Apache Flink 应用程序创建和运行 Kinesis Data Analytics

在本练习中，您创建一个 Kinesis Data Analytics 应用程序，它将 Amazon MSK 主题作为源，并将 Amazon S3 存储桶作为接收器。

本节包含以下步骤。

创建相关资源

在为本练习创建针对 Apache Flink 的 Kinesis Data Analytics 应用程序之前，请创建以下相关资源：

- 基于 Amazon VPC 和 Amazon MSK 集群的虚拟私有云 (VPC)
- 存储应用程序代码和输出的 Amazon S3 存储桶 (`ka-app-<username>`)

创建 VPC 和 Amazon MSK 群集

要创建 VPC 和 Amazon MSK 集群以从 Kinesis Data Analytics 应用程序中访问，请执行入门使用 Amazon MSK教程。

在完成本教程时，请注意以下几点：

- 记录集群的引导服务器列表。您可以使用以下命令获取引导服务器列表，替换*ClusterArn*使用 MSK 集群的 Amazon 资源名称 (ARN)：

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
    "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094"
}
```

- 在执行教程中的步骤时，请务必使用所选的Amazon代码、命令和控制台条目中的区域。

创建 Amazon S3 存储桶

您可以使用控制台创建 Amazon S3 存储桶。有关创建此资源的说明，请参阅以下主题：

- 如何创建 S3 存储桶？中的Amazon Simple Storage Service 用户指南. 附加您的登录名，以便为 Amazon S3 存储桶指定全局唯一的名称，例如**ka-app-*<username>***.

其他资源

当您创建应用程序时，Kinesis Data Analytics 会创建以下亚马逊CloudWatch资源（如果不存在）

- 名为 `/aws/kinesis-analytics-java/MyApplication` 的日志组。
- 名为 `kinesis-analytics-log-stream` 的日志流。

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入 Amazon MSK 主题，以供应用程序处理。

1. Connect 到您在中创建的客户端实例步骤 4: 创建客户端计算机的开始使用亚马逊 MSK教程。
2. 安装 Python3、Pip 和 Kafka Python 库：

```
$ sudo yum install python37
$ curl -O https://bootstrap.pypa.io/get-pip.py
$ python3 get-pip.py --user
$ pip install kafka-python
```

3. 使用以下内容创建名为 `stock.py` 的文件。替换`BROKERS`您之前记录的引导经纪商名单的价值。

```
from kafka import KafkaProducer
import json
import random
from datetime import datetime

BROKERS = "<Bootstrap Brokers List>"
producer = KafkaProducer(
    bootstrap_servers=BROKERS,
    value_serializer=lambda v: json.dumps(v).encode('utf-8'),
    retry_backoff_ms=500,
    request_timeout_ms=20000,
    security_protocol='PLAINTEXT')


def getStock():
    data = {}
    now = datetime.now()
    str_now = now.strftime("%Y-%m-%d %H:%M:%S")
    data['event_time'] = str_now
    data['ticker'] = random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV'])
    price = random.random() * 100
    data['price'] = round(price, 2)
    return data


while True:
    data =getStock()
    # print(data)
    try:
        future = producer.send("AWSKafkaTutorialTopic", value=data)
        producer.flush()
        record_metadata = future.get(timeout=10)
        print("sent event to Kafka! topic {} partition {} offset
 {}".format(record_metadata.topic, record_metadata.partition, record_metadata.offset))
    except Exception as e:
        print(e.with_traceback())
```

4. 在本教程的后面部分，您运行 `stock.py` 脚本，以将数据发送到应用程序。

```
$ python3 stock.py
```

下载并检查 Apache Flink 流式处理 Java 代码

此示例的 Java 应用程序代码可从GitHub.

要下载 Java 应用程序代码

1. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

2. 导航到 amazon-kinesis-data-analytics-java-examples/GettingStartedTable目录。

请注意有关应用程序代码的以下信息：

- 一个项目对象模型 (pom.xml)文件包含有关应用程序的配置和依赖关系（包括 Kinesis Data Analytics 库）的信息。
- StreamingJob.java 文件包含定义应用程序功能的 main 方法。
- 该应用程序使用FlinkKafkaConsumer阅读亚马逊 MSK 主题。以下代码段创建FlinkKafkaConsumer对象：

```
final FlinkKafkaConsumer<StockRecord> consumer = new
 FlinkKafkaConsumer<StockRecord>(kafkaTopic, new KafkaEventDeserializationSchema(),
 kafkaProps);
```

- 您的应用程序使用创建源和接收连接器以访问外部资源StreamExecutionEnvironment和TableEnvironment对象。
- 应用程序使用动态应用程序属性创建源连接器和接收器连接器，因此您可以在不重新编译代码的情况下指定应用程序参数（如 S3 存储桶）。

```
//read the parameters from the Kinesis Analytics environment
Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
Properties flinkProperties = null;

String kafkaTopic = parameter.get("kafka-topic", "AWSKafkaTutorialTopic");
String brokers = parameter.get("brokers", "");
String s3Path = parameter.get("s3Path", "");

if (applicationProperties != null) {
    flinkProperties = applicationProperties.get("FlinkApplicationProperties");
}

if (flinkProperties != null) {
    kafkaTopic = flinkProperties.get("kafka-topic").toString();
    brokers = flinkProperties.get("brokers").toString();
    s3Path = flinkProperties.get("s3Path").toString();
}
```

有关运行时属性的更多信息，请参阅运行时属性 (p. 18)。

编译应用程序代码

在本节中，您使用 Apache Maven 编译器创建应用程序的 Java 代码。有关安装 Apache Maven 和 Java 开发工具包 (JDK) 的信息，请参阅完成练习的先决条件 (p. 76)。

编译应用程序代码

1. 要使用您的应用程序代码，您将其编译和打包成 JAR 文件。您可以通过两种方式之一编译和打包您的代码：

   - 使用命令行 Maven 工具。在包含 `pom.xml` 文件的目录中通过运行以下命令创建您的 JAR 文件：

     ```
     mvn package -Dflink.version=1.13.2
     ```

   - 设置开发环境。有关详细信息，请参阅您的开发环境文档。

     Note

     提供的源代码依赖于 Java 11 中的库。

   您可以作为 JAR 文件上传您的包，也可以将包压缩为 ZIP 文件并上传。如果您使用 Amazon CLI 创建应用程序，您可以指定您的代码内容类型（JAR 或 ZIP）。
2. 如果编译时出错，请验证 `JAVA_HOME` 环境变量设置正确。

如果应用程序成功编译，则创建以下文件：

`target/aws-kinesis-analytics-java-apps-1.0.jar`

上传 Apache Flink 流式处理 Java 代码

在本节中，您将创建 Amazon S3 存储桶并上传应用程序代码。

上传应用程序代码

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. Enter**`ka-app-code-<username>`**中的Bucket name字段中返回的子位置类型。将后缀（如您的用户名）添加到存储桶名称，以使其具有全局唯一性。请选择 Next (下一步)。
4. 在配置选项步骤中，让设置保持原样，然后选择下一步。
5. 在设置权限步骤中，让设置保持原样，然后选择下一步。
6. 请选择 Create bucket (创建存储桶)。
7. 在 Amazon S3 控制台中，选择ka-app-code-***`<username>`***存储桶，然后选择上传.
8. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `aws-kinesis-analytics-java-apps-1.0.jar` 文件。请选择 Next (下一步)。
9. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在 Amazon S3 存储桶中，应用程序可以在其中访问代码。

创建和运行 Kinesis Data Analytics 应用程序

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **MyApplication**。
   - 对于描述，输入 **My java test app**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。
   - 将版本保留为Apache Flink 1.13.2 (推荐版本).
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
5. 选择 Create application（ 创建应用程序 ）。

   Note

   使用控制台创建 Kinesis Data Analytics 应用程序时，您可以选择为您的应用程序创建 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源使用您的应用程序名称和区域命名，如下所示：
   - 策略：kinesis-analytics-service-*MyApplication*-*us-west-2*
   - 角色：kinesis-analytics-*MyApplication*-*us-west-2*

编辑 IAM 策略

编辑 IAM 策略以添加权限，以访问 Amazon S3 存储桶。

编辑 IAM 策略以添加 S3 存储桶权限

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择策略。选择控制台在上一部分中为您创建的 **kinesis-analytics-service-MyApplication-us-west-2** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。
4. 将以下策略示例中突出显示的部分添加到策略中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
```

```
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "WriteObjects",
            "Effect": "Allow",
            "Action": [
                "s3:Abort*",
                "s3:DeleteObject*",
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-<username>",
                "arn:aws:s3:::ka-app-<username>/*"
            ]
        }
    ]
}
```

配置应用程序

可以使用以下过程配置应用程序。

要配置应用程序

1. 在存储库的MyApplication在页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**ka-app-code-*<username>***.
   - 适用于指向 Amazon S3 对象的路径输入**aws-kinesis-analytics-java-apps-1.0.jar**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。

4. UNDER属性，选择创建组. 适用于Group ID (组 ID)输入**FlinkApplicationProperties**.
5. 输入以下应用程序属性和值：

密钥	值
kafka-topic	**AWSKafkaTutorialTopic**
brokers	***Your Amazon MSK cluster's Bootstrap Brokers list***
s3Path	**ka-app-*<username>***
security.protocol	**SSL**
ssl.truststore.location	**/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts**
ssl.truststore.password	**changeit**

6. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
7. 适用于CloudWatch记录中，选择启用”复选框。
8. 在Virtual Private Cloud (VPC)选择部分，选择基于 Amazon MSK 集群的 VPC 配置. 选择AWSKafkaTutorialCluster.
9. 选择 Update (更新) 。

Note

当您选择启用亚马逊时CloudWatchKinesis Data Analytics 为您创建日志组和日志流。这些资源的名称如下所示：

- 日志组：`/aws/kinesis-analytics/MyApplication`
- 日志流：`kinesis-analytics-log-stream`

运行应用程序

可以使用以下过程运行应用程序。

要运行应用程序

1. 在存储库的MyApplication在页面上，选择运行. 确认该操作。
2. 当应用程序正在运行时，请刷新页面。控制台将显示 Application graph (应用程序图表)。
3. 在您的 Amazon EC2 客户端中，运行之前创建的 Python 脚本，将记录写入 Amazon MSK 集群以便您的应用程序处理：

```
$ python3 stock.py
```

停止应用程序

要停止应用程序，请在MyApplication在页面上，选择停止. 确认该操作。

下一个步骤

清理 Amazon 资源 (p. 101)

清理 Amazon 资源

本节包含清理过程Amazon在入门 (表 API) 教程中创建的资源。

本主题包含以下部分。

删除 Kinesis Data Analytics 应用程序

请使用以下过程删除应用程序。

删除应用程序

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 在应用程序页面上，选择Delete然后确认删除。

删除您的亚马逊 MSK 集群

要删除您的亚马逊 MSK 集群，请按照步骤 8：删除 Amazon MSK 群集中的Amazon Managed Streaming for Apache Kafka 开发人员指南.

删除 VPC

要删除 Amazon VPC，请执行以下操作：

- 打开 Amazon VPC 控制台。
- 选择你的 VPC。
- 对于 Actions (操作)，请选择 Delete VPC (删除 VPC)。

删除 Amazon S3 对象和存储桶

可以使用以下过程删除 S3 对象和存储桶。

删除 S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择ka-app-code-*<username>*存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除 IAM 资源

可以使用以下过程删除 IAM 资源。

删除 IAM 资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-***<your-region>***政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏中，选择 Roles（角色）。
7. 选择kinesis-analyticsMyApplication-***<your-region>***角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除 CloudWatch 资源

可以使用以下过程删除CloudWatch资源的费用。

删除CloudWatch资源

1. 打开 CloudWatch 控制台 (https://console.aws.amazon.com/cloudwatch/)。
2. 在导航栏中，选择 Logs (日志)。
3. 选择/aws/kinesis-analytics/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

下一个步骤

后续步骤 (p. 102)

后续步骤

现在您已创建并运行使用表 API 的 Kinesis Data Analytics 应用程序，请参阅第 5 步：后续步骤 (p. 91)中的使用 Amazon Kinesis Data Analytics Apache Flink 入门 (DataStreamAPI) (p. 76).

Amazon Kinesis Data Analytics for Python Flink for P

本节介绍了使用 Python 和 Table API 的适用于 Apache Flink 的 Kinesis Data Analytics 的基本概念。它介绍了可用于创建和测试应用程序的选项。它还提供了相应的说明以安装所需的工具，以完成本指南中的教程和创建第一个应用程序。

主题

Pyflink 入门——适用于 Apache 的 Python 解释器 | Amazon Web Services 科技

在开始之前，我们建议您首先观看以下视频：

Pyflink 入门——适用于 Apache 的 Python 解释器 | Amazon Web Services 科技

适用于 Flink 应用程序的 Kinesis Data Analytics 组件

为了处理数据，Kinesis Data Analytics 运行时使用 Python 应用程序处理输入和生成输出。

Kinesis Data Analytics 应用程序具有以下组件：

- 运行时属性：您可以使用运行时属性配置应用程序，而无需重新编译应用程序代码。
- 表格来源：应用程序通过源使用数据。一个资源连接器从 Kinesis Data Streams、Amazon MSK 主题或类似主题中读取数据。有关更多信息，请参阅 表 API 来源 (p. 13)。
- 函数：应用程序使用一个或多个功能以处理数据。一个功能可以转换、丰富或聚合数据。
- 接收器：应用程序使用接收器将生成的数据发送到外部源。一个下沉连接器将数据写入 Kinesis 数据流、Kinesis Data Firehose 传输流、亚马逊 MSK 主题、Amazon S3 存储桶等。有关更多信息，请参阅 表 API 接收器 (p. 14)。

在创建和打包应用程序代码后，您可以将代码包上传到 Amazon S3 存储桶。然后，您创建 Kinesis Data Analytics 应用程序。您在代码包位置中传入流数据源，它通常是接收应用程序处理的数据的流或文件位置。

先决条件

在开始本教程之前，请先完成使用 Amazon Kinesis Data Analytics Apache Flink 入门 (DataStreamAPI) (p. 76)：

- 第 1 步：设置Amazon创建管理员用户 (p. 77)
- 第 2 步：设置 Amazon Command Line Interface (Amazon CLI) (p. 79)

要了解其用法，请参阅 创建应用程序 (p. 104)。

创建并运行适用于 Python 的 Kinesis Data Analytics 应用程序

在本练习中，您将使用 Kinesis 流作为源和接收器来创建适用于 Python 应用程序的 Kinesis Data Analytics 应用程序。

本节包含以下步骤。

创建相关资源

在为本练习创建适用于 Apache Flink 的 Kinesis Data Analytics 应用程序之前，请创建以下相关资源：

- 两个 Kinesis 流用于输入和输出。
- Amazon S3 存储段，用于存储应用程序代码和输出 (`ka-app-<username>`)

创建两个 Kinesis Streams

对于本练习 Kinesis Data Analytics 应用程序，请创建两个 Kinesis Data Analytics (`ExampleInputStream`和`ExampleOutputStream`)。您的应用程序将这些数据流用于应用程序源和目标流。

您可以使用 Amazon Kinesis 控制台或以下控制台创建这些流。Amazon CLI命令。有关控制台说明，请参阅创建和更新数据流中的Amazon Kinesis Data Streams.

创建数据流 (Amazon CLI)

1. 创建第一个流 (`ExampleInputStream`)，使用以下Amazon Kinesis`create-stream` Amazon CLI命令。

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. 要创建应用程序用来写入输出的第二个流，请运行同一命令（将流名称更改为 `ExampleOutputStream` ）。

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

创建 AAmazon S3 存储桶

您可以使用控制台创建 Amazon S3 存储段。有关创建此资源的说明，请参阅以下主题：

- 如何创建 S3 存储桶？中的Amazon Simple Service. 附加您的登录名，以便为 Amazon S3 存储桶指定全局唯一的名称，例如**ka-app-*<username>***.

其他资源

当您创建应用程序时，Kinesis Data Analytics 将创建以下亚马逊 CloudWatch 资源（如果不存在）：

- 名为 `/aws/kinesis-analytics-java/MyApplication` 的日志组。
- 名为 `kinesis-analytics-log-stream` 的日志流。

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入流，以供应用程序处理。

> **Note**
>
> 此部分需要 Amazon SDK for Python (Boto)。

> **Note**
>
> 本节中的 Python 脚本使用Amazon CLI. 你必须配置你的Amazon CLI使用您的账户凭证和默认区域。配置 CourceAmazon CLI，输入以下内容：

```
aws configure
```

1. 使用以下内容创建名为 `stock.py` 的文件：

```
import datetime
import json
import random
import boto3
import time

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")
        time.sleep(2)
```

```
if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

2. 运行 stock.py 脚本：

```
$ python stock.py
```

在完成本教程的其余部分时，请将脚本保持运行状态。

创建和检查 Apache Flink 流式处理 Python 代码

本示例的 Python 应用程序代码可从 GitHub. 要下载应用程序代码，请执行以下操作：

1. 如果尚未安装 Git 客户端，请安装它。有关更多信息，请参阅安装 Git。
2. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

3. 导航到 amazon-kinesis-data-analytics-java-examples/python/GettingStarted目录。

应用程序代码位于 streaming-file-sink.py 文件中。请注意有关应用程序代码的以下信息：

- 应用程序使用 Kinesis 表源从源流中进行读取。以下片段调用create_table函数来创建 Kinesis 表源：

```
table_env.execute_sql(
        create_table(input_table_name, input_stream, input_region, stream_initpos)
    )
```

这些区域有：create_table函数使用 SQL 命令创建由流式处理源支持的表：

```
def create_table(table_name, stream_name, region, stream_initpos):
    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time TIMESTAMP(3),
                WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND

              )
              PARTITIONED BY (ticker)
              WITH (
                'connector' = 'kinesis',
                'stream' = '{1}',
                'aws.region' = '{2}',
                'scan.stream.initpos' = '{3}',
                'sink.partitioner-field-delimiter' = ';',
                'sink.producer.collection-max-count' = '100',
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(
        table_name, stream_name, region, stream_initpos
    )
```

- 应用程序创建两个表，然后将一个表的内容写入另一个表。

```
    # 2. Creates a source table from a Kinesis Data Stream
    table_env.execute_sql(
        create_table(input_table_name, input_stream, input_region, stream_initpos)
```

```
)

# 3. Creates a sink table writing to a Kinesis Data Stream
table_env.execute_sql(
    create_table(output_table_name, output_stream, output_region, stream_initpos)
)

# 4. Inserts the source table data into the sink table
table_result = table_env.execute_sql("INSERT INTO {0} SELECT * FROM {1}"
                     .format(output_table_name, input_table_name))
```

- 该应用程序使用 Flink 连接器，从flink- sql-connector-kinesis_1.13.2文件。
- 使用第三方 python 软件包时（例如boto3），则需要将添加到中GettingStarted文件夹在哪里`getting-started.py`位于中。无需在 Apache Flink 或 Kinesis Data Analytics 中添加任何其他配置。可以在以下网站找到示例：如何在 PyfLink 中使用 boto3.

上传 Apache Flink 流式处理 Python 代码

在本节中，您将创建 Amazon S3 存储桶并上传应用程序代码。

使用控制台上传应用程序代码：

1. 使用您首选的压缩应用程序来压缩`getting-started.py`和https://mvnrepository.com/artifact/org.apache.flink/flink- sql-connector-kinesis_1.13.2文件。命名存档`myapp.zip`. 如果在档案中包含外部文件夹，则必须将其包含在配置文件中的代码的路径中：`GettingStarted/getting-started.py`.
2. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
3. 选择 Create bucket (创建存储桶)。
4. Enter**`ka-app-code-<username>`**中的Bucket name字段中返回的子位置类型。将后缀（如您的用户名）添加到存储桶名称，以使其具有全局唯一性。选择 Next（下一步）。
5. 在配置选项步骤中，让设置保持原样，然后选择下一步。
6. 在设置权限步骤中，让设置保持原样，然后选择下一步。
7. 请选择 Create bucket (创建存储桶)。
8. 在 Amazon S3 控制台中，选择ka-app-code-***`<username>`***存储桶，然后选择上传.
9. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `myapp.zip` 文件。选择 Next（下一步）。
10. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

使用上传应用程序代码Amazon CLI：

> **Note**
>
> 不要使用 Finder (macOS) 或 Windows 资源管理器 (Windows) 中的压缩功能来创建myapp.zip存档。这可能会导致应用程序代码无效。

1. 使用您首选的压缩应用程序来压缩`getting-started.py`和https://mvnrepository.com/artifact/org.apache.flink/flink- sql-connector-kinesis_1.13.2文件。命名存档`myapp.zip`. 如果在档案中包含外部文件夹，则必须将其包含在配置文件中的代码的路径中：`GettingStarted/getting-started.py`.
2. 运行以下 命令：

```
$ aws s3 --region aws region cp myapp.zip s3://ka-app-code-<username>
```

您的应用程序代码现在存储在 Amazon S3 存储桶中，应用程序可以在其中访问代码。

创建和运行 Kinesis Data Analytics 应用程序

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用程序.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于描述，输入 **`My java test app`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。
   - 将版本保留为Apache Flink 版本 1.13.1（推荐版本）.
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
5. 选择 Create application（创建应用程序）。

   Note

   在使用控制台创建 Kinesis Data Analytics 应用程序时，您可以选择为应用程序创建 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源是使用您的应用程序名称和区域命名的，如下所示：
   - 策略：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
   - 角色：`kinesis-analytics-`*`MyApplication`*`-`*`us-west-2`*

配置应用程序

可以使用以下过程配置该应用程序。

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶，输入**`ka-app-code-`**.
   - 适用于Amazon S3 对象的路径，输入**`myapp.zip`**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
4. UNDER属性，选择添加组. 适用于Group ID (组 ID)，输入**`consumer.config.0`**.
5. 输入以下应用程序属性和值：

密钥	值
`input.stream.name`	**`ExampleInputStream`**
`aws.region`	**`us-west-2`**
`flink.stream.initpos`	**`LATEST`**

选择Save（保存）。

6. UNDER属性，选择添加组。适用于Group ID (组 ID)，输入**producer.config.0**.
7. 输入以下应用程序属性和值：

密钥	值
output.stream.name	**ExampleOutputStream**
aws.region	**us-west-2**
shard.count	**1**

8. UNDER属性，选择添加组。适用于Group ID (组 ID)，输入**kinesis.analytics.flink.run.options**. 这个特殊的属性组告诉你的应用程序在哪里可以找到它的代码资源。有关更多信息，请参阅 指定代码文件 (p. 17)。
9. 输入以下应用程序属性和值：

密钥	值
python	**getting-started.py**
jarfile	**flink-sql-connector-kinesis_2.12-1.13.1.jar**

10. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
11. 适用于CloudWatch 记录，选择启用”复选框。
12. 选择 Update（更新）。

Note

当您选择启用亚马逊时 CloudWatch 日志记录，Kinesis Data Analytics 将为您创建日志组和日志流。这些资源的名称如下所示：

- 日志组：`/aws/kinesis-analytics/MyApplication`
- 日志流：`kinesis-analytics-log-stream`

编辑 IAM 策略

编辑 IAM 策略以添加权限以访问 Amazon S3 存储桶。

编辑 IAM 策略以添加 S3 存储桶权限

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择策略。选择控制台在上一部分中为您创建的 **kinesis-analytics-service-MyApplication-us-west-2** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。
4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*012345678901*) 替换为您的账户 ID。

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

```
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/myapp.zip"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

运行应用程序

通过运行应用程序，打开 Apache Flink 控制面板，然后选择所需的 Flink 作业，可以查看 Flink 作业图。

停止应用程序

要停止应用程序，在MyApplication页面上，选择停止. 确认该操作。

下一个步骤

清理 Amazon 资源 (p. 111)

清理 Amazon 资源

本节包含清理过程。Amazon在入门 (Python) 教程中创建的资源。

本主题包含以下部分。

删除 Kinesis Data Analytics 应用程序

请使用以下过程删除该应用程序。

删除应用程序

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 在应用程序页面上，选择Delete然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream.
3. 在ExampleInputStream页面上，选择删除 Kinesis Streams然后确认删除。
4. 在Kinesis Streams页面上，选择ExampleOutputStream，选择操作，选择Delete，然后确认删除。

删除Amamazon S3 对象和存储桶

可以使用以下过程删除 S3 对象和存储桶。

删除 S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择ka-app-code-***<username>***存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除您的 IAM 资源

可以使用以下过程删除您的 IAM 资源。

删除您的 IAM 资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-***<your-region>***政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏中，选择 Roles（角色）。
7. 选择kinesis-analyticsMyApplication-***<your-region>***角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除您的 CloudWatch 资源

可以使用以下过程删除您的 CloudWatch 资源的费用。

删除 CloudWatch 资源

1. 打开 CloudWatch 控制台位于https://console.aws.amazon.com/cloudwatch/.
2. 在导航栏中，选择 Logs (日志)。
3. 选择/kinesis-analytics/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

使用 Apache Beam 创建 Kinesis Data Analytics 应用程序

您可以使用Apache Beam框架与 Kinesis Data Analytics 应用程序一起处理流数据。使用 Apache Beam 的 Kinesis Data Analytics 应用程序Apache Flink 跑步者来执行 Beam 管道。

有关如何在 Kinesis Data Analytics 应用程序中使用 Apache Beam 的教程，请参阅使用CloudFormation使用 Kinesis Data Analytics (p. 113).

本主题包含下列部分：

将 Apache Beam 与 Kinesis Data Analytics 结合使用

请注意以下关于将 Apache Flink 运行器与 Kinesis Data Analytics 结合使用的事项：

- 在 Kinesis Data Analytics 控制台中无法查看 Apache Beam 指标。
- 只有使用 Apache Flink 1.8 及更高版本的 Kinesis Data Analytics 应用程序才支持 Apache Beam。使用 Apache Flink 版本 1.6 的 Kinesis Data Analytics 应用程序不支持 Apache Beam。

Beam 功能

Kinesis Data Analytics 支持与 Apache Flink 跑步者相同的 Apache Beam 功能。有关 Apache Flink 运行器支持哪些功能的信息，请参阅Beam 兼容性矩阵.

我们建议您在 Kinesis Data Analytics 服务中测试 Apache Flink 应用程序，以验证我们支持您的应用程序所需的所有功能。

使用 Apache Beam 创建应用程序

在本练习中，您将创建一个 Kinesis Data Analytics 应用程序，该应用程序使用Apache Beam. Apache Beam 是一种用于处理流数据的编程模型。有关将 Apache Beam 与 Kinesis Data Analytics 结合使用的信息，请参阅使用 Apache Beam (p. 113).

> Note
>
> 要为本练习设置所需的先决条件，请先完成入门 (DataStreamAPI) (p. 76)练习。

本主题包含下列部分：

创建相关资源

在为本练习创建 Kinesis Data Analytics 应用程序之前，请创建以下相关资源：

- 两个 Kinesis 数据流 (`ExampleInputStream`和`ExampleOutputStream`)
- 用于存储应用程序代码的 Amazon S3 存储桶 (`ka-app-code-<username>`)

您可以使用控制台创建 Kinesis 流和 Amazon S3 存储桶。有关创建这些资源的说明，请参阅以下主题：

- 创建和更新数据流中的Amazon Kinesis Data Streams 开发人员指南. 将数据流命名为 **`ExampleInputStream`** 和 **`ExampleOutputStream`**。
- 如何创建 S3 存储桶？中的Amazon Simple Storage Service 用户指南. 通过附加您的登录名，为 Amazon S3 存储桶指定全局唯一的名称，例如**`ka-app-code-<username>`**.

将示例记录写入输入流

在本节中，您使用 Python 脚本将随机字符串写入流，以供应用程序处理。

> **Note**
>
> 此部分需要 Amazon SDK for Python (Boto)。

1. 使用以下内容创建名为 `ping.py` 的文件：

```
import json
import boto3
import random

kinesis = boto3.client('kinesis')

while True:
        data = random.choice(['ping', 'telnet', 'ftp', 'tracert', 'netstat'])
        print(data)
        kinesis.put_record(
                StreamName="ExampleInputStream",
                Data=data,
                PartitionKey="partitionkey")
```

2. 运行 `ping.py` 脚本：

```
$ python ping.py
```

在完成本教程的其余部分时，请将脚本保持运行状态。

下载并检查应用程序代码

在 GitHub 中提供了该示例的 Java 应用程序代码。要下载应用程序代码，请执行以下操作：

1. 如果尚未安装 Git 客户端，请安装它。有关更多信息，请参阅安装 Git。
2. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

3. 导航到 amazon-kinesis-data-analytics-java-examples/Beam目录。

应用程序代码位于 BasicBeamStreamingJob.java 文件中。请注意有关应用程序代码的以下信息：

- 该应用程序使用 Apache BeamParDo通过调用名为的自定义转换函数来处理传入记录PingPongFn.

  要调用的代码PingPongFn函数如下所示：

```
.apply("Pong transform",
    ParDo.of(new PingPongFn())
```

- 使用 Apache Beam 的 Kinesis Data Analytics 应用程序需要以下组件。如果你没有将这些组件和版本包含在你的pom.xml，您的应用程序从环境依赖项加载不正确的版本，并且由于版本不匹配，因此应用程序在运行时崩溃。

```
<jackson.version>2.10.2</jackson.version>
...
<dependency>
    <groupId>com.fasterxml.jackson.module</groupId>
    <artifactId>jackson-module-jaxb-annotations</artifactId>
    <version>2.10.2</version>
</dependency>
```

- 这些区域有：PingPongFn除非输入数据是，否则转换函数将输入数据传递到输出流中Ping，在这种情况下它发出字符串Pong\ n转到输出流。

  转换函数的代码如下：

```
    private static class PingPongFn extends DoFn<KinesisRecord, byte[]> {
    private static final Logger LOG = LoggerFactory.getLogger(PingPongFn.class);

    @ProcessElement
    public void processElement(ProcessContext c) {
        String content = new String(c.element().getDataAsBytes(),
 StandardCharsets.UTF_8);
        if (content.trim().equalsIgnoreCase("ping")) {
            LOG.info("Ponged!");
            c.output("pong\n".getBytes(StandardCharsets.UTF_8));
        } else {
            LOG.info("No action for: " + content);
            c.output(c.element().getDataAsBytes());
        }
    }
}
```

编译应用程序代码

要编译应用程序，请执行以下操作：

1. 如果还没有 Java 和 Maven，请安装它们。有关更多信息，请参阅入门 (DataStreamAPI) (p. 76)教程中的先决条件 (p. 76)。
2. 使用以下命令编译应用程序：

```
mvn package -Dflink.version=1.13.2 -Dflink.version.minor=1.8
```

Note

提供的源代码依赖于 Java 11 中的库。

编译应用程序将创建应用程序 JAR 文件 (`target/basic-beam-app-1.0.jar`)。

上传 Apache Flink 流式处理 Java 代码

在本节中，您将应用程序代码上传到在创建相关资源 (p. 114)部分。

1. 在 Amazon S3 控制台中，选择ka-app 代码***<username>***存储桶，然后选择上传.
2. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `basic-beam-app-1.0.jar` 文件。
3. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在 Amazon S3 存储桶中，应用程序可以在其中访问代码。

创建并运行 Kinesis Data Analytics 应用程序

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。

     Note

     Kinesis Data Analytics 使用 Apache Flink 版本 1.13.2。
   - 将版本下拉列表保留为Apache Flink 版本 1.13.2 (推荐版本) .
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
5. 选择 Create application (创建应用程序) 。

   Note

   在使用控制台创建 Kinesis Data Analytics 应用程序时，您可以选择为应用程序创建 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。使用您的应用程序名称和区域命名这些 IAM 资源，如下所示：

   - 策略：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
   - 角色：`kinesis-analytics-`*`MyApplication`*`-`*`us-west-2`*

编辑 IAM 策略

编辑 IAM 策略以添加访问 Kinesis 数据流的权限。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择策略。选择控制台在上一部分中为您创建的 **kinesis-analytics-service-MyApplication-us-west-2** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。
4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/basic-beam-app-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

配置应用程序

1. 在 MyApplication (我的应用程序) 页面上，选择 Configure (配置)。
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**ka-app-code-*<username>***.
   - 适用于Amazon S3 对象的路径输入**basic-beam-app-1.0.jar**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
4. 在 Properties (属性) 下，对于 Group ID (组 ID)，输入 **BeamApplicationProperties**。
5. 输入以下应用程序属性和值：

密钥	值
InputStreamName	**ExampleInputStream**
OutputStreamName	**ExampleOutputStream**
AwsRegion	**us-west-2**

6. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
7. 对于 CloudWatch logging (CloudWatch 日志记录)，选中启用复选框。
8. 选择 Update (更新) 。

Note

当你选择启用 CloudWatch 记录中，Kinesis Data Analytics 为您创建日志组和日志流。这些资源的名称如下所示：

- 日志组：`/aws/kinesis-analytics/MyApplication`
- 日志流：`kinesis-analytics-log-stream`

该日志流用于监控应用程序。这与应用程序用于发送结果的日志流不同。

运行应用程序

可以通过运行应用程序、打开 Apache Flink 控制面板并选择所需的 Flink 作业来查看 Flink 作业图。

你可以查看 Kinesis Data Analytics 指标 CloudWatch 控制台，以验证应用程序是否正常运行。

清理 Amazon 资源

本节包含清理在滚动窗口教程中创建的 Amazon 资源的过程。

本主题包含下列部分：

删除 Kinesis Data Analytics 应用程序

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 在应用程序的页面中，选择 Delete (删除)，然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Streams 面板中，选择示例输入流.
3. 在 ExampleInputStream 页面中，选择 Delete Kinesis Stream，然后确认删除。
4. 在 Kinesis streams (Kinesis 流) 页面中，选择 ExampleOutputStream，选择 Actions (操作)，选择 Delete (删除)，然后确认删除。

删除您的 Amazon S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择 ka-app-code-***<username>*** 存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除 IAM 资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择 kinesis-analytics-service-MyApplication-***<####>*** 策略。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏中，选择 Roles（角色）。
7. 选择 kinesis-analytics-MyApplication-***<####>*** 角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除 CloudWatch 资源

1. 打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/.
2. 在导航栏中，选择 Logs (日志)。
3. 选择 /aws/kinesis-analytics/MyApplication 日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

后续步骤

现在，您已经创建并运行了使用 Apache Beam 转换数据的基本 Kinesis Data Analytics 应用程序，请参阅以下应用程序了解更高级的 Kinesis Data Analytics 解决方案示例。

- Beam on Kinesis Data Analytics 直播研讨会：在本次研讨会中，我们探讨了一个端到端的示例，将批处理和流媒体方面结合在一个统一的 Apache Beam 管道中。

培训研讨会、实验室和解决方案实施

以下 end-to-end 示例展示了高级 Kinesis Data Analytics 解决方案

主题

在将 Apache Flink 应用程序部署到 Kinesis Data Analytics（Apache Flink）

本次研讨会将向您展示在本地启动和开始开发 Apache Flink 应用程序的基础知识，其长期目标是部署到适用于 Apache Flink 的 Kinesis Data Analytics。

可在此处找到解决方案：使用 Apache Flink 进行本地发展的入门指南

使用 Kinesis Data Analytics Studio 检测事件

本研讨会介绍了使用 Kinesis Data Analytics 工作室进行的事件检测并将其部署为 Kinesis Data Analytics 应用

可在此处找到解决方案：Apache Flink 使用 Kinesis Data Analytics 进行事件检测

AmazonAmazon Kinesis 的流数据解决方案

这些区域有：Amazon适用于 Amazon Kinesis 的流数据解决方案自动配置Amazon轻松捕获、存储、处理和交付流数据所需的服务。该解决方案为解决流式数据使用案例提供了多种选择。Kinesis Data Analytics 选项提供了 end-to-end 直播 ETL 示例演示了对模拟的纽约出租车数据运行分析操作的真实世界应用程序。

每个解决方案包含以下组件：

- 一个Amazon CloudFormation用于部署完整示例的软件包。
- 一个 CloudWatch 显示应用程序指标的仪表板。
- CloudWatch 对最相关的应用程序指标发出警报。
- 所有必要的 IAM 角色和策略。

可在此处找到解决方案：Amazon Kinesis 的流数据解决方案

点击流实验室与 Apache Flink 和 Apache Kafka

一个端到端的实验室，该实验室用于点击流使用案例，使用适用于 Apache Kafka 的亚马逊托管流媒体进行流媒体存储和适用于 Apache Flink 的 Amazon Kinesis Data Analytics 应用程序用于流

可在此处找到解决方案：Clickstream Lab

使用应用程序 Auto Scaling 定制

一个示例，可帮助用户使用应用程序 Auto Scaling 自动扩展 Apache Flink 应用程序的 Kinesis Data Analytics。这使用户能够设置自定义扩展策略和自定义扩展属性。

可在此处找到解决方案：Kinesis Data Analytics ing Flink App Auto Scaling

亚马逊 CloudWatch 控制面板

示例 CloudWatch 控制面板，用于监控 Amazon Kinesis Data Analytics 应用 示例仪表板还包括演示应用程序以帮助演示仪表板的功能。

可在此处找到解决方案：Kinesis Data Analytics 指标仪表板

AmazonAmazon MSK 流式处理方案

这些区域有：Amazon亚马逊 MSK 流数据解决方案提供Amazon CloudFormation数据流经生产者、流式存储、消费者和目的地的模板。

可在此处找到解决方案：AmazonAmazon MSK 流式处理方案

GitHub 上的更多 Kinesis Data Analytics 解决

以下 end-to-end 示例演示了高级 Kinesis Data Analytics 解决方案，可在 GitHub 上获得

- Amazon Kinesis Data Analytics Flink — 基准测试实用程序
- Apache Flink 快照管理器 — Amazon Kinesis Data Analytics
- 使用 Apache Flink 和 Amazon Kinesis Data Analytics 进行流式处理
- 对客户反馈进行实时情绪分析

实用程序

以下实用程序可以使 Flink 版 Kinesis Data Analytics 服务更易于使用：

主题

快照管理器

Flink 应用程序的最佳做法是定期触发保存点/快照，以实现更加无缝的故障恢复。快照管理器可自动执行此任务，具备以下优势：

- 为 Apache Flink 应用程序拍摄正在运行的 Kinesis Data Analytics
- 获取应用程序快照计数
- 检查计数是否超过所需的快照数量
- 删除超过所需数量的旧快照

有关示例，请参阅 。Flink 的快照管理器

基准

Kinesis Data Analytics Flink 基准实用程序可帮助针对 Apache Flink 应用程序进行容量规划、集成测试和基准测试。

有关示例，请参阅 。基准

Kinesis Data Analytics 示例

此部分提供在 Amazon Kinesis Data Analytics 中创建和使用应用程序的示例。它们包括示例代码和 step-by-step 说明以帮助您创建 Kinesis Data Analytics 应用程序和测试结果。

在分析这些示例之前，我们建议您先查看以下内容：

- 工作方式 (p. 2)
- 入门 (DataStreamAPI) (p. 76)

> **Note**
>
> 这些示例假定您使用美国西部（俄勒冈）区域 (`us-west-2`)。如果您使用不同的区域，请相应地更新应用程序代码、命令和 IAM 角色。

主题

DataStream API 示例

以下示例演示如何使用 Apache Flink 创建应用程序 DataStream API。

主题

示例：滚动窗口

在本练习中，您创建一个使用滚动窗口聚合数据的 Kinesis Data Analytics 应用程序。在 Flink 中，该功能状态。要禁用终止保护，请使用以下内容：

```
sink.producer.aggregation-enabled' = 'false'
```

> **Note**
>
> 要为本练习设置所需的先决条件，请先完成入门 (DataStreamAPI) (p. 76)练习。

本主题包含下列部分：

创建相关资源

在为本练习创建 Kinesis Data Analytics 应用程序之前，请创建以下相关资源：

- 两个 Kinesis 数据流 (`ExampleInputStream`和`ExampleOutputStream`)
- Amazon S3 存储桶 (`ka-app-code-<username>`)

您可以使用控制台创建 Kinesis 流和 Amazon S3 存储桶。有关创建这些资源的说明，请参阅以下主题：

- 创建和更新数据流中的Amazon Kinesis Data Streams 开发者指南. 将数据流命名为 **`ExampleInputStream`** 和 **`ExampleOutputStream`**。
- 如何创建 S3 存储桶？中的Amazon Storage Service. 附加您的登录名，以便为 Amazon S3 存储桶指定全局唯一的名称，例如**`ka-app-code-<username>`**.

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入流，以供应用程序处理。

> **Note**
>
> 此部分需要 Amazon SDK for Python (Boto)。

1. 使用以下内容创建名为 `stock.py` 的文件：

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")
```

```
if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

2. 运行 stock.py 脚本：

```
$ python stock.py
```

在完成本教程的其余部分时，请将脚本保持运行状态。

下载并检查应用程序代码

该示例的 Java 应用程序代码可从 GitHub. 要下载应用程序代码，请执行以下操作：

1. 如果尚未安装 Git 客户端，请安装它。有关更多信息，请参阅安装 Git。
2. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

3. 导航到 amazon-kinesis-data-analytics-java-examples/TumblingWindow目录。

应用程序代码位于 TumblingWindowStreamingJob.java 文件中。请注意有关应用程序代码的以下信息：

- 应用程序使用 Kinesis 源从源流中进行读取。以下代码段创建了 Kinesis source：

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- 添加以下 import 语句：

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13
```

- 应用程序使用 timeWindow 操作符在 5 秒的滚动窗口中查找每个股票代号的值计数。以下代码创建操作符，并将聚合的数据发送到新的 Kinesis Data Streams 接收器：

```
input.flatMap(new Tokenizer()) // Tokenizer for generating words
                .keyBy(0) // Logically partition the stream for each word
                //.timeWindow(Time.seconds(5)) // Tumbling window definition (Flink 1.11)
  .window(TumblingProcessingTimeWindows.of(Time.seconds(5))) //Flink 1.13
                .sum(1) // Sum the number of words per partition
                .map(value -> value.f0 + "," + value.f1.toString() + "\n")
                .addSink(createSinkFromStaticConfig());
```

编译应用程序代码

要编译应用程序，请执行以下操作：

1. 如果还没有 Java 和 Maven，请安装它们。有关更多信息，请参阅入门 (DataStreamAPI) (p. 76)教程中的先决条件 (p. 76)。
2. 使用以下命令编译应用程序：

```
mvn package -Dflink.version=1.13.2
```

Note

提供的源代码依赖于 Java 11 中的库。

编译应用程序将创建应用程序 JAR 文件 (`target/aws-kinesis-analytics-java-apps-1.0.jar`)。

上传 Apache Flink 流式处理 Java 代码

在本节中，您将应用程序代码上传到在一节中创建的 Amazon S3 存储桶创建相关资源 (p. 124)部分。

1. 在 Amazon S3 控制台中，选择ka-app-code-***<username>***存储桶，然后选择上传.
2. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `aws-kinesis-analytics-java-apps-1.0.jar` 文件。
3. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在存储Amazon S3 中，应用程序可以在其中访问代码。

创建和运行 Kinesis Data Analytics 应用程序

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用程序.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。

     Note

     Kinesis Data Analytics 使用 Apache Flink 版本 1.132。
   - 将版本下拉菜单保留为Apache Flink 1.13.2（推荐版本）.
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
5. 选择 Create application（创建应用程序）。

   Note

   在使用控制台创建 Kinesis Data Analts 应用程序时，您可以选择为应用程序创建一个 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源是使用您的应用程序名称和区域命名的，如下所示：

   - 策略：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
   - 角色：`kinesis-analytics-`*`MyApplication`*`-`*`us-west-2`*

编辑 IAM 策略

编辑 IAM 策略添加访问数据流数据流的Kinesis 限限限。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。

2. 选择策略。选择控制台在上一部分中为您创建的 **kinesis-analytics-service-MyApplication-us-west-2** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。
4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**ka-app-code-*<username>***.
   - 适用于Amazon S3 对象的路径输入**aws-kinesis-analytics-java-apps-1.0.jar**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
4. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
5. 适用于CloudWatch 记录，选择启用”复选框。
6. 选择 Update (更新) 。

   Note

   当你选择启用 CloudWatch 日志组和日Kinesis Data Analytics 和日志组日志组和日志志组日志组和日志志组日志 这些资源的名称如下所示：

   - 日志组：`/aws/kinesis-analytics/MyApplication`
   - 日志流：`kinesis-analytics-log-stream`

   该日志流用于监控应用程序。这与应用程序用于发送结果的日志流不同。

运行应用程序

1. 在存储库的MyApplication页面上，选择运行. 离开了在没有快照的情况下运行选项，然后确认操作。
2. 当应用程序正在运行时，请刷新页面。控制台将显示 Application graph (应用程序图表)。

你可以在上查看 Kinesis Data Analytics 指标 CloudWatch 控制台，以验证应用程序是否正常工作。

清理 Amazon 资源

本节包含清理在滚动窗口教程中创建的 Amazon 资源的过程。

本主题包含下列部分：

删除 Kinesis Data Analytics 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 在应用程序的页面中，选择 Delete (删除)，然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream.
3. 在ExampleInputStream页面上，选择删除 Kinesis Streams然后确认删除。
4. 在Kinesis Streams页面上，选择ExampleOutputStream，选择操作，选择Delete，然后确认删除。

删除您的 Amazon S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择ka-app-code-***<username>***存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-***<your-region>***政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏中，选择 Roles（角色）。
7. 选择kinesines-anticticMyApplication-***<your-region>***角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除日期和时间 CloudWatch 资源

1. 打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/.
2. 在导航栏中，选择 Logs (日志)。
3. 选择/aws/kinesis-analytictics/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

示例：滑动窗口

在本练习中，您创建一个使用滑动窗口聚合数据的 Kinesis Data Analytics 应用程序。在 Flink 中，该功能状态。要禁用终止保护，请使用以下内容：

```
sink.producer.aggregation-enabled' = 'false'
```

> **Note**
>
> 要为本练习设置所需的先决条件，请先完成入门 (DataStreamAPI) (p. 76)练习。

本主题包含下列部分：

- 创建相关资源 (p. 130)
- 将示例记录写入输入流 (p. 130)
- 下载并检查应用程序代码 (p. 131)

创建相关资源

在为本练习创建 Kinesis Data Analytics 应用程序之前，请创建以下相关资源：

- 两个 Kinesis 数据流（`ExampleInputStream` 和 `ExampleOutputStream`）。
- Amazon S3 存储桶 (`ka-app-code-<username>`)

您可以使用控制台创建 Kinesis 流和 Amazon S3 存储桶。有关创建这些资源的说明，请参阅以下主题：

- 创建和更新数据流中的Amazon Kinesis Data Streams 开发者指南. 将数据流命名为 **`ExampleInputStream`** 和 **`ExampleOutputStream`**。
- 如何创建 S3 存储桶？中的Amazon Storage Service. 附加您的登录名，以便为 Amazon S3 存储桶指定全局唯一的名称，例如**`ka-app-code-<username>`**.

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入流，以供应用程序处理。

> **Note**
>
> 此部分需要 Amazon SDK for Python (Boto)。

1. 使用以下内容创建名为 `stock.py` 的文件：

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

2. 运行 stock.py 脚本：

```
$ python stock.py
```

在完成本教程的其余部分时，请将脚本保持运行状态。

下载并检查应用程序代码

该示例的 Java 应用程序代码可从 GitHub. 要下载应用程序代码，请执行以下操作：

1. 如果尚未安装 Git 客户端，请安装它。有关更多信息，请参阅安装 Git。
2. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

3. 导航到 amazon-kinesis-data-analytics-java-examples/SlidingWindow目录。

应用程序代码位于 SlidingWindowStreamingJobWithParallelism.java 文件中。请注意有关应用程序代码的以下信息：

- 应用程序使用 Kinesis 源从源流中进行读取。以下代码段创建了 Kinesis source：

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- 应用程序使用 timeWindow 操作符在 10 秒的滑动窗口（以 5 秒为增量）中查找每个股票代号的最小值。以下代码创建操作符，并将聚合的数据发送到新的 Kinesis Data Streams 接收器：
- 添加以下 import 语句：

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13
```

- 应用程序使用 timeWindow 操作符在 5 秒的滚动窗口中查找每个股票代号的值计数。以下代码创建操作符，并将聚合的数据发送到新的 Kinesis Data Streams 接收器：

```
input.flatMap(new Tokenizer()) // Tokenizer for generating words
                .keyBy(0) // Logically partition the stream for each word
                //.timeWindow(Time.seconds(5)) // Tumbling window definition (Flink 1.11)
  .window(TumblingProcessingTimeWindows.of(Time.seconds(5))) //Flink 1.13
                .sum(1) // Sum the number of words per partition
                .map(value -> value.f0 + "," + value.f1.toString() + "\n")
                .addSink(createSinkFromStaticConfig());
```

编译应用程序代码

要编译应用程序，请执行以下操作：

1. 如果还没有 Java 和 Maven，请安装它们。有关更多信息，请参阅入门 (DataStreamAPI) (p. 76)教程中的先决条件 (p. 76)。
2. 使用以下命令编译应用程序：

```
mvn package -Dflink.version=1.13.2
```

Note

提供的源代码依赖于 Java 11 中的库。

编译应用程序将创建应用程序 JAR 文件 (`target/aws-kinesis-analytics-java-apps-1.0.jar`)。

上传 Apache Flink 流式处理 Java 代码

在本节中，您将应用程序代码上传到在一节中创建的 Amazon S3 存储桶创建相关资源 (p. 130)部分。

1. 在 Amazon S3 控制台中，选择ka-app-code-***<username>***存储桶，然后选择上传.
2. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `aws-kinesis-analytics-java-apps-1.0.jar` 文件。
3. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在存储Amazon S3 中，应用程序可以在其中访问代码。

创建和运行 Kinesis Data Analytics 应用程序

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用程序.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。
   - 将版本下拉菜单保留为Apache Flink 1.13.2 (推荐版本) .
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
5. 选择 Create application (创建应用程序) 。

Note

在使用控制台创建 Kinesis Data Analts 应用程序时，您可以选择为应用程序创建一个 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源是使用您的应用程序名称和区域命名的，如下所示：

- 策略：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
- 角色：`kinesis-analytics-`*`MyApplication`*`-`*`us-west-2`*

编辑 IAM 策略

编辑 IAM 策略添加访问数据流数据流的Kinesis 限限限。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择策略。选择控制台在上一部分中为您创建的 **`kinesis-analytics-service-MyApplication-us-west-2`** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。

4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**ka-app-code-*<username>***.
   - 适用于Amazon S3 对象的路径输入**aws-kinesis-analytics-java-apps-1.0.jar**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
4. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
5. 适用于CloudWatch 记录，选择启用”复选框。
6. 选择 Update（更新）。

   Note

   当您选择启用亚马逊时 CloudWatch 日志组和日Kinesis Data Analytics 和日志组日志组和日志志组日志组和日志志组日志 这些资源的名称如下所示：

   - 日志组：`/aws/kinesis-analytics/MyApplication`
   - 日志流：`kinesis-analytics-log-stream`

   该日志流用于监控应用程序。这与应用程序用于发送结果的日志流不同。

配置应用程序并行度

该应用程序示例使用任务的并行执行功能。以下应用程序代码设置 `min` 操作符的并行度：

```
.setParallelism(3) // Set parallelism for the min operator
```

应用程序并行度不能大于预置的并行度（默认为 1）。要增加应用程序的并行度，请使用以下 Amazon CLI 操作：

```
aws kinesisanalyticsv2 update-application
      --application-name MyApplication
      --current-application-version-id <VersionId>
      --application-configuration-update "{\"FlinkApplicationConfigurationUpdate\":
 { \"ParallelismConfigurationUpdate\": {\"ParallelismUpdate\": 5, \"ConfigurationTypeUpdate
\": \"CUSTOM\" }}}"
```

您可以使用检索当前应用程序版本 IDDescribeApplication要么ListApplications行动。

运行应用程序

通过运行应用程序，打开 Apache Flink 控制面板，然后选择所需的 Flink 作业，可以查看 Flink 作业图。

你可以在上查看 Kinesis Data Analytics 指标 CloudWatch 控制台来验证应用程序是否正常工作。

清理 Amazon 资源

本节包含清理在滑动窗口教程中创建的 Amazon 资源的过程。

本主题包含下列部分：

- 删除 Kinesis Data Analytics 应用程序 (p. 135)

删除 Kinesis Data Analytics 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 在应用程序的页面中，选择 Delete (删除)，然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream.
3. 在ExampleInputStream页面上，选择删除 Kinesis Streams然后确认删除。
4. 在Kinesis Streams页面上，选择ExampleOutputStream，选择操作，选择Delete，然后确认删除。

删除您的 Amazon S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择ka-app-code-***<username>***存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-***<your-region>***政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏中，选择 Roles（角色）。
7. 选择kinesines-anticticMyApplication-***<your-region>***角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除日期和时间 CloudWatch 资源

1. 打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/.
2. 在导航栏中，选择 Logs (日志)。
3. 选择/aws/kinesis-analytictics/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

示例：写入 Amazon S3 存储桶

在本练习中，您创建适用于 Apache Flink 的 Kinesis Data Analytics 应用程序，它将Kinesis 数据流作为源，并将Amazon S3 存储桶作为接收器。通过使用接收器，您可以在 Amazon S3 控制台中验证应用程序输出。

Note

要为本练习设置所需的先决条件，请先完成入门 (DataStreamAPI) (p. 76)练习。

本主题包含下列部分：

创建相关资源

在为本练习创建Kinesis Data Analytics for Apache For Apache Flink 应用程序之前，请创建以下相关资源：

- Kinesis 数据流 (`ExampleInputStream`)。
- 用于存储应用程序代码和输出的 Amazon S3 存储段 (`ka-app-<username>`)

Note

在 Kinesis Data Analytics 上启用了服务器端加密的情况下，适用于 Apache 的 Kinesis Data Analytics Flink 无法将数据写入 Amazon S3。

您可以使用控制台创建 Kinesis 流和 Amazon S3 存储桶。有关创建这些资源的说明，请参阅以下主题：

- 创建和更新数据流中的Amazon Kinesis Data Streams 开发者指南. 将数据流命名为 **`ExampleInputStream`**。
- 如何创建 S3 存储桶？中的Amazon Storage Service. 附加您的登录名，以便为 Amazon S3 存储桶指定全局唯一的名称，例如**`ka-app-<username>`**. 创建两个文件夹 (**`code`**和**`data`**) 在Amazon S3 存储桶中。

该应用程序将创建以下内容 CloudWatch 资源（如果不存在）：

- 名为 `/aws/kinesis-analytics-java/MyApplication` 的日志组。
- 名为 `kinesis-analytics-log-stream` 的日志流。

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入流，以供应用程序处理。

Note

此部分需要 Amazon SDK for Python (Boto)。

1. 使用以下内容创建名为 `stock.py` 的文件：

```
import datetime
import json
```

```
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

2. 运行 `stock.py` 脚本：

```
$ python stock.py
```

在完成本教程的其余部分时，请将脚本保持运行状态。

下载并检查应用程序代码

该示例的 Java 应用程序代码可从 GitHub. 要下载应用程序代码，请执行以下操作：

1. 如果尚未安装 Git 客户端，请安装它。有关更多信息，请参阅安装 Git。
2. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

3. 导航到 `amazon-kinesis-data-analytics-java-examples/S3Sink`目录。

应用程序代码位于 `S3StreamingSinkJob.java` 文件中。请注意有关应用程序代码的以下信息：

- 应用程序使用 Kinesis 源从源流中进行读取。以下代码段创建了 Kinesis source：

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- 您需要添加以下导入语句：

```
import org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows;
```

- 应用程序使用 Apache Flink S3 接收器写入到 Amazon S3。

  接收器在滚动窗口中读取消息，将消息编码为 S3 存储桶对象，然后将编码的对象发送到 S3 接收器。以下代码将对象进行编码以发送到 Amazon S3：

```
input.map(value -> { // Parse the JSON
            JsonNode jsonNode = jsonParser.readValue(value, JsonNode.class);
            return new Tuple2><jsonNode.get("TICKER").toString(), 1);
        }).returns(Types.TUPLE(Types.STRING, Types.INT))
                .keyBy(0) // Logically partition the stream for each word
                // .timeWindow(Time.minutes(1)) // Tumbling window definition // Flink
 1.11
                .window(TumblingProcessingTimeWindows.of(Time.minutes(1))) // Flink 1.13
                .sum(1) // Count the appearances by ticker per partition
                .map(value -> value.f0 + " count: " + value.f1.toString() + "\n")
                .addSink(createS3SinkFromStaticConfig());

        env.execute("Flink S3 Streaming Sink Job");
```

Note

应用程序使用 Flink`StreamingFileSink`要写入 Amazon S3 的对象。有关的更多信息`StreamingFileSink`，请参阅StreamingFileSink中的Flink 文档.

修改应用程序代码

在本节中，您修改应用程序代码以将输出写入到 Amazon S3 存储桶。

使用您的用户名更新以下行以指定应用程序的输出位置：

```
private static final String s3SinkPath = "s3a://ka-app-<username>/data";
```

编译应用程序代码

要编译应用程序，请执行以下操作：

1. 如果还没有 Java 和 Maven，请安装它们。有关更多信息，请参阅入门 (DataStreamAPI) (p. 76)教程中的先决条件 (p. 76)。
2. 使用以下命令编译应用程序：

```
mvn package -Dflink.version=1.13.2
```

编译应用程序将创建应用程序 JAR 文件 (`target/aws-kinesis-analytics-java-apps-1.0.jar`)。

Note

提供的源代码依赖于 Java 11 中的库。

上传 Apache Flink 流式处理 Java 代码

在本节中，您将应用程序代码上传到在一节中创建的 Amazon S3 存储桶创建相关资源 (p. 136)部分。

1. 在 Amazon S3 控制台中，选择ka-app-***<username>***存储桶，导航到代码文件夹，然后选择 Core上传.
2. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `aws-kinesis-analytics-java-apps-1.0.jar` 文件。
3. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在存储Amazon S3 中，应用程序可以在其中访问代码。

创建和运行 Kinesis Data Analytics 应用程序

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用程序.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。
   - 将版本下拉菜单保留为Apache Flink 1.13.2（推荐版本）.
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
5. 选择 Create application（创建应用程序）。

   Note

   在使用控制台创建 Kinesis Data Analts 应用程序时，您可以选择为应用程序创建一个 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源是使用您的应用程序名称和区域命名的，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。
   - 将版本保留为Apache Flink 1.13.2（推荐版本）.
6. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
7. 选择 Create application（创建应用程序）。

   Note

   在使用控制台创建适用于 Apache Flink 的 Kinesis Data Analts Flink 应用程序时，您可以选择为应用程序创建一个 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源是使用您的应用程序名称和区域命名的，如下所示：
   - 策略：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
   - 角色：`kinesis-analytics-`*`MyApplication`*`-`*`us-west-2`*

编辑 IAM 策略

编辑 IAM 策略添加访问数据流数据流的Kinesis 限限限。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择策略。选择控制台在上一部分中为您创建的 **`kinesis-analytics-service-MyApplication-us-west-2`** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。
4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*`012345678901`*) 替换为您的账户 ID。替<username>换为您的用户名。

```
{
```

```
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::kinesis-analytics-placeholder-s3-bucket/kinesis-
analytics-placeholder-s3-object"
            ]
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:*"
            ]
        },
        {
            "Sid": "ListCloudwatchLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:%LOG_GROUP_PLACEHOLDER%:log-
stream:*"
            ]
        },
        {
            "Sid": "PutCloudwatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:%LOG_GROUP_PLACEHOLDER%:log-
stream:%LOG_STREAM_PLACEHOLDER%"
            ]
        }
        ,
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteObjects",
            "Effect": "Allow",
            "Action": [
                "s3:Abort*",
                "s3:DeleteObject*",
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-<username>",
                "arn:aws:s3:::ka-app-<username>/*"
```

```
            ]
        }
    ]
}
```

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**ka-app-*<username>***.
   - 适用于Amazon S3 对象的路径输入**code/aws-kinesis-analytics-java-apps-1.0.jar**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
4. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
5. 适用于CloudWatch 记录，选择启用”复选框。
6. 选择 Update (更新) 。

   Note

   当你选择启用 CloudWatch 日志组和日Kinesis Data Analytics 和日志组日志组和日志志组日志组和日志志组日志 这些资源的名称如下所示：

   - 日志组：`/aws/kinesis-analytics/MyApplication`
   - 日志流：`kinesis-analytics-log-stream`

   该日志流用于监控应用程序。这与应用程序用于发送结果的日志流不同。

运行应用程序

1. 在存储库的MyApplication页面上，选择运行. 离开了在没有快照的情况下运行选项，然后确认操作。
2. 当应用程序正在运行时，请刷新页面。控制台将显示 Application graph (应用程序图表)。

验证应用程序输出

在 Amazon S3 控制台中，打开数据C3 存储桶中的文件夹。

几分钟后，将显示包含来自应用程序的聚合数据的对象。

Note

在 Flink 中，该功能状态。要禁用终止保护，请使用以下内容：

```
sink.producer.aggregation-enabled' = 'false'
```

可选：自定义源和接收器

在本节中，您将自定义源对象和汇对象的设置。

Note

更改以下各节中描述的代码部分后，执行以下操作以重新加载应用程序代码：

- 重复以下步骤the section called “编译应用程序代码” (p. 138)部分来编译更新的应用程序代码。
- 重复以下步骤the section called “上传 Apache Flink 流式处理 Java 代码” (p. 138)部分上传更新的应用程序代码。
- 在控制台中的应用程序页面上，选择配置然后选择更新将更新的应用程序代码重新加载到您的应用程序中。

本部分包含以下部分：

配置数据分区

在本部分中，您将配置流式文件接收器在 S3 存储桶中创建的文件夹的名称。您可以通过向流式文件接收器添加存储桶分配器来实现此目的。

要自定义在 S3 存储桶中创建的文件夹名称：

1. 将以下导入语句添加至开头`S3StreamingSinkJob.java`file:

```
import
 org.apache.flink.streaming.api.functions.sink.filesystem.rollingpolicies.DefaultRollingPolicy;
import
 org.apache.flink.streaming.api.functions.sink.filesystem.bucketassigners.DateTimeBucketAssigner;
```

2. 更新`createS3SinkFromStaticConfig()`方法看起来与以下内容类似：

```
private static StreamingFileSink<String> createS3SinkFromStaticConfig() {

    final StreamingFileSink<String> sink = StreamingFileSink
        .forRowFormat(new Path(s3SinkPath), new SimpleStringEncoder<String>("UTF-8"))
        .withBucketAssigner(new DateTimeBucketAssigner("yyyy-MM-dd--HH"))
        .withRollingPolicy(DefaultRollingPolicy.create().build())
        .build();
    return sink;
}
```

前面的代码示例使用`DateTimeBucketAssigner`使用自定义日期格式在 S3 存储桶中创建文件夹。这些区域有：`DateTimeBucketAssigner`使用当前系统时间创建存储桶名称。如果要创建自定义存储桶分配器以进一步自定义创建的文件夹名称，则可以创建一个类来实现BucketAssigner. 你可以通过使用来实现你的自定义逻辑`getBucketId`方法。

的自定义实现`BucketAssigner`可使用上下文参数获取有关记录的更多信息，以确定其目标文件夹。

配置读取Fream

在本节中，您配置在源流上进行读取的频率。

默认情况下，Kinesis Streams 使用器每秒从源流中读取五次。如果有多个客户端从流中读取，或者应用程序需要重试读取记录，则此频率将导致问题。您可以通过设置消费者的读取频率来避免这些问题。

要设置 Kinesis 消费者的读取频率，请将SHARD_GETRECORDS_INTERVAL_MILLIS设置。

下面的代码示例将SHARD_GETRECORDS_INTERVAL_MILLIS设置为一秒：

```
kinesisConsumerConfig.setProperty(ConsumerConfigConstants.SHARD_GETRECORDS_INTERVAL_MILLIS,
 "1000");
```

配置写入缓冲区

在本节中，您将配置接收器的写入频率和其他设置。

默认情况下，应用程序每分钟向目标存储桶写入一次。您可以更改此间隔和其他设置，方法是配置DefaultRollingPolicy对象。

> **Note**
>
> 每次应用程序创建检查点时，Apache Flink 流式文件接收器都会写入其输出存储桶。默认情况下，应用程序每分钟创建一个检查点。要增加 S3 接收器的写入间隔，还必须增加检查点间隔。

配置DefaultRollingPolicy对象中，执行以下操作：

1. 增加应用程序的CheckpointInterval设置。以下输入 UpdateApplication操作将检查点间隔设置为 10 分钟:

```
{
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "ConfigurationTypeUpdate" : "CUSTOM",
            "CheckpointIntervalUpdate": 600000
         }
      }
   },
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5
}
```

   要使用上述代码，请指定当前应用程序版本。您可以使用ListApplicationsaction.

2. 将以下导入语句添加至开头S3StreamingSinkJob.javafile:

```
import java.util.concurrent.TimeUnit;
```

3. 更新createS3SinkFromStaticConfig中的方法S3StreamingSinkJob.java文件与以下内容类似：

```
private static StreamingFileSink<String> createS3SinkFromStaticConfig() {

        final StreamingFileSink<String> sink = StreamingFileSink
                .forRowFormat(new Path(s3SinkPath), new
 SimpleStringEncoder<String>("UTF-8"))
                .withBucketAssigner(new DateTimeBucketAssigner("yyyy-MM-dd--HH"))
                .withRollingPolicy(
                        DefaultRollingPolicy.create()
                            .withRolloverInterval(TimeUnit.MINUTES.toMillis(8))
                            .withInactivityInterval(TimeUnit.MINUTES.toMillis(5))
                            .withMaxPartSize(1024 * 1024 * 1024)
                            .build())
                .build();
        return sink;
```

```
    }
```

前面的代码示例将写入 Amazon S3 存储桶的频率设置为 8 分钟。

有关配置 Apache Flink 流文件接收器的更多信息，请参阅行编码格式中的Flink 文档.

清理 Amazon 资源

本节包含清理过程Amazon您在 Amazon S3 教程中创建的资源。

本主题包含下列部分：

删除 Kinesis Data Analytics 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 在应用程序页面上，选择Delete然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream.
3. 在存储库的ExampleInputStream页面上，选择删除 Kinesis Streams然后确认删除。

删除Amazon S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择 ka-app-***<username>*** 存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏上，选择策略.
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-***<your-region>***政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏上，选择角色.
7. 选择kinesines-anticticMyApplication-***<your-region>***角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除日期和时间 CloudWatch 资源

1. 打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/.
2. 在导航栏上，选择日志.
3. 选择/aws/kinesis-analytictics/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

教程：使用 Kinesis Data Analytics 应用程序将数据从一个 MSK 集群复制到 VPC 中的另一个集群

以下教程说明了如何创建具有 Amazon MSK 集群和两个主题的 Amazon VPC，以及如何创建从一个 Amazon MSK 主题中读取并写入到另一个主题的 Kinesis Data Analts 应用程序。

> Note
>
> 要为本练习设置所需的先决条件，请先完成入门 (DataStreamAPI) (p. 76)练习。

本教程包含以下部分：

使用 Amazon MSK 集群创建Amazon VPC

要创建示例 VPC 和 Amazon MSK 集群以从 Kinesis Data Analytics 应用程序中访问，请按照使用Amazon MSK 入门教程。

在完成本教程时，请注意以下几点：

- In第 5 步：创建主题，重复`kafka-topics.sh --create`命令以创建名为的目标主题`AWSKafkaTutorialTopicDestination`：

```
bin/kafka-topics.sh --create --zookeeper ZooKeeperConnectionString --replication-factor 3
 --partitions 1 --topic AmazonKafkaTutorialTopicDestination
```

- 记录集群的引导服务器列表。您可以使用以下命令获取引导服务器列表（替换*ClusterArn*使用您的 MSK 集群的 ARN）：

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
    "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094"
}
```

- 在执行教程中的步骤时，请务必使用所选的Amazon代码、命令和控制台条目中的区域。

创建应用程序代码

在本节中，您下载并编译应用程序 JAR 文件。我们建议使用 Java 11。

该示例的 Java 应用程序代码可从 GitHub. 要下载应用程序代码，请执行以下操作：

1. 如果尚未安装 Git 客户端，请安装它。有关更多信息，请参阅安装 Git。
2. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

3. 应用程序代码位于 `amazon-kinesis-data-analytics-java-examples/KafkaConnectors/KafkaGettingStartedJob.java` 文件中。您可以检查代码以熟悉 Kinesis Data Analytics 应用程序代码的结构。
4. 使用命令行 Maven 工具或首选的开发环境以创建 JAR 文件。要使用命令行 Maven 工具编译 JAR 文件，请输入以下内容：

```
mvn package -Dflink.version=1.13.2
```

如果构建成功，则会创建以下文件：

```
target/KafkaGettingStartedJob-1.0.jar
```

> **Note**
>
> 提供的源代码依赖于 Java 11 中的库。如果你使用的是开发环境，

上传 Apache Flink 流式处理 Java 代码

在本节中，您将应用程序代码上传到在一节中创建的 Amazon S3 存储桶入门 (DataStreamAPI) (p. 76)教程。

> **Note**
>
> 如果您从入门教程中删除了 Amazon S3 存储桶，请按照the section called “上传 Apache Flink 流式处理 Java 代码” (p. 82)再次进行。

1. 在 Amazon S3 控制台中，选择ka-app-code-***<username>***存储桶，然后选择上传.
2. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `KafkaGettingStartedJob-1.0.jar` 文件。
3. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在存储Amazon S3 中，应用程序可以在其中访问代码。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Amazon Kinesis Data Analytics 控制面板创建分析应用程序.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 适用于运行时，选择Flink 版本 1.13.2.

4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
5. 选择 Create application（创建应用程序）。

Note

在使用控制台创建 Kinesis Data Analts 应用程序时，您可以选择为应用程序创建一个 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源是使用您的应用程序名称和区域命名的，如下所示：

- 策略：kinesis-analytics-service-*MyApplication*-*us-west-2*
- 角色：kinesis-analytics-*MyApplication*-*us-west-2*

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**ka-app-code-*<username>***.
   - 适用于Amazon S3 对象的路径输入**KafkaGettingStartedJob-1.0.jar**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。

   Note

   使用控制台指定应用程序资源时（例如 CloudWatch 日志或 Amazon VPC）时，控制台修改应用程序执行角色以授予访问这些资源的权限。
4. 在 Properties (属性) 下面，选择 Add Group (添加组)。使用以下属性创建一个名为 **KafkaSource** 的属性组：

密钥	值
topic	AmazonKafkaTutorialTopic
bootstrap.servers	*############*
security.protocol	SSL
ssl.truststore.location	/usr/usr/usr/security/security/security/security/securityts
ssl.truststore.password	changeit

Note

默认证书的 ssl.truststore.password 为“changeit”；如果使用默认证书，则不需要更改该值。

再次选择 Add Group (添加组)。使用以下属性创建一个名为 **KafkaSink** 的属性组：

密钥	值
topic	AmazonKafkaTutorialTopicDestination

密钥	值
bootstrap.servers	###########
security.protocol	SSL
ssl.truststore.location	/usr/usr/usr/security/security/security/security/securityts
ssl.truststore.password	changeit
交易.timeout.ms	1000

应用程序代码读取上述应用程序属性，以配置用于与 VPC 和 Amazon MSK 集群交互的源和接收器。有关使用属性的更多信息，请参阅运行时属性 (p. 18)。

5. 在 Snapshots (快照) 下面，选择 Disable (禁用)。这样，就可以轻松更新应用程序，而无需加载无效的应用程序状态数据。
6. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
7. 适用于CloudWatch 记录，选择启用”复选框。
8. 在 Virtual Private Cloud (VPC) 部分中，选择要与应用程序关联的 VPC。选择与您的 VPC 关联的子网和安全组，您希望应用程序使用它们访问 VPC 资源。
9. 选择 Update (更新) 。

Note

当你选择启用 CloudWatch 日志组和日Kinesis Data Analytics 和日志组日志组和日志志组日志组和日志志组日志 这些资源的名称如下所示：

- 日志组：`/aws/kinesis-analytics/MyApplication`
- 日志流：`kinesis-analytics-log-stream`

该日志流用于监控应用程序。

运行应用程序

通过运行应用程序，打开 Apache Flink 控制面板，然后选择所需的 Flink 作业，可以查看 Flink 作业图。

测试应用程序

在本节中，您将记录写入到源主题。应用程序从源主题中读取记录，并将其写入到目标主题中。您可以将记录写入到源主题以及从目标主题中读取记录，以验证应用程序是否正常工作。

要写入和读取主题中的记录，请按照中的步骤操作步骤 6：生成和使用数据中的使用Amazon MSK 入门教程。

要从目标主题中读取，请在到集群的第二个连接中使用目标主题名称，而不是源主题：

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --consumer.config
 client.properties --topic AmazonKafkaTutorialTopicDestination --from-beginning
```

如果在目标主题中没有任何记录，请参阅故障排除 (p. 316)主题中的无法访问 VPC 中的资源 (p. 319)一节。

示例：将 EFO 使用者与 Kinesis Data Streams 配合使用

在本练习中，您将创建一个 Kinesis Data Analytics 应用程序，该应用程序使用增强型扇出功能 (EFO)使用者。如果 Kinesis 使用者使用 EFO，Kinesis Data Streams 服务会为其提供自己的专用带宽，而不是让使用者与从流中读取数据的其他使用者共享流的固定带宽。

有关将 EFO 与 Kinesis 使用者结合使用的更多信息，请参阅FLINUX 针对 Kinesis 消费者的增强型扇出功能.

您在本示例中创建的应用程序使用AmazonKinesis 连接器 (flink-connector-kinesis) 1.13.2。

> Note
>
> 要为本练习设置所需的先决条件，请先完成入门 (DataStreamAPI) (p. 76)练习。

本主题包含下列部分：

创建相关资源

在为本练习创建 Kinesis Data Analytics 应用程序之前，请创建以下相关资源：

- 两个 Kinesis 数据流 (`ExampleInputStream`和`ExampleOutputStream`)
- Amazon S3 存储桶 (`ka-app-code-<username>`)

您可以使用控制台创建 Kinesis 流和 Amazon S3 存储桶。有关创建这些资源的说明，请参阅以下主题：

- 创建和更新数据流中的Amazon Kinesis Data Streams 开发者指南. 将数据流命名为 **`ExampleInputStream`** 和 **`ExampleOutputStream`**。
- 如何创建 S3 存储桶？中的Amazon Storage Service. 附加您的登录名，以便为 Amazon S3 存储桶指定全局唯一的名称，例如**`ka-app-code-<username>`**.

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入流，以供应用程序处理。

> Note
>
> 此部分需要 Amazon SDK for Python (Boto)。

1. 使用以下内容创建名为 `stock.py` 的文件：

```
import datetime
import json
import random
import boto3
```

```
STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

2. 运行 `stock.py` 脚本：

```
$ python stock.py
```

在完成本教程的其余部分时，请将脚本保持运行状态。

下载并检查应用程序代码

该示例的 Java 应用程序代码可从 GitHub. 要下载应用程序代码，请执行以下操作：

1. 如果尚未安装 Git 客户端，请安装它。有关更多信息，请参阅安装 Git。
2. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

3. 导航到 `amazon-kinesis-data-analytics-java-examples/EfoConsumer`目录。

应用程序代码位于 `EfoApplication.java` 文件中。请注意有关应用程序代码的以下信息：

- 您可以通过在 Kinesis 使用器上设置以下参数来启用 EFO 使用者：
  - 记录_PUBLISHER_TYPE: 将该参数设置为EFO让您的应用程序使用 EFO 使用者访问 Kinesis 数据流数据。
  - EFO_CONSUMER_NAME: 将此参数设置为在此流的使用者中唯一的字符串值。在同一 Kinesis Data Stream 中重复使用使用者名称将导致以前使用该名称的使用者被终止。
- 以下代码示例演示如何将值分配给使用者配置属性以使用 EFO 使用者从源流中读取：

```
consumerConfig.putIfAbsent(RECORD_PUBLISHER_TYPE, "EFO");
consumerConfig.putIfAbsent(EFO_CONSUMER_NAME, "basic-efo-flink-app");
```

编译应用程序代码

要编译应用程序，请执行以下操作：

1. 如果还没有 Java 和 Maven，请安装它们。有关更多信息，请参阅入门 (DataStreamAPI) (p. 76)教程中的先决条件 (p. 76)。
2. 使用以下命令编译应用程序：

```
mvn package -Dflink.version=1.13.2
```

Note

提供的源代码依赖于 Java 11 中的库。

编译应用程序将创建应用程序 JAR 文件 (`target/aws-kinesis-analytics-java-apps-1.0.jar`)。

上传 Apache Flink 流式处理 Java 代码

在本节中，您将应用程序代码上传到在一节中创建的 Amazon S3 存储桶创建相关资源 (p. 149)部分。

1. 在 Amazon S3 控制台中，选择ka-app-code-***<username>***存储桶，然后选择上传.
2. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `aws-kinesis-analytics-java-apps-1.0.jar` 文件。
3. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在存储Amazon S3 中，应用程序可以在其中访问代码。

创建和运行 Kinesis Data Analytics 应用程序

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用程序.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。

     Note

     Kinesis Data Analytics 使用 Apache Flink 版本 1.132。
   - 将版本下拉菜单保留为Apache Flink 1.13.2（推荐版本）.
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
5. 选择 Create application（创建应用程序）。

   Note

   在使用控制台创建 Kinesis Data Analts 应用程序时，您可以选择为应用程序创建一个 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源是使用您的应用程序名称和区域命名的，如下所示：

   - 策略：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*

- 角色：kinesis-analytics-*MyApplication*-*us-west-2*

编辑 IAM 策略

编辑 IAM 策略添加访问数据流数据流的Kinesis 限限限。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择策略。选择控制台在上一部分中为您创建的 **kinesis-analytics-service-MyApplication-us-west-2** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。
4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*012345678901*) 替换为您的账户 ID。

   Note

   这些权限授予应用程序访问 EFO 消费者的能力。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "AllStreams",
            "Effect": "Allow",
            "Action": [
```

```
                "kinesis:ListShards",
                "kinesis:ListStreamConsumers",
                "kinesis:DescribeStreamSummary"
            ],
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/*"
        },
        {
            "Sid": "Stream",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:RegisterStreamConsumer",
                "kinesis:DeregisterStreamConsumer"
            ],
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        },
        {
            "Sid": "Consumer",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStreamConsumer",
                "kinesis:SubscribeToShard"
            ],
            "Resource": [
                "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream/
consumer/my-flink-efo-consumer",
                "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream/
consumer/my-flink-efo-consumer:*"
            ]
        }
    ]
}
```

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**ka-app-code-*<username>***.
   - 适用于Amazon S3 对象的路径输入**aws-kinesis-analytics-java-apps-1.0.jar**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
4. UNDER属性，选择创建组. 适用于Group ID (组 ID)输入**ConsumerConfigProperties**.
5. 输入以下应用程序属性和值：

密钥	值
flink.stream.recordpublisher	**EFO**
flink.stream.efo.consumername	**my-flink-efo-consumer**

密钥	值
INPUT_STREAM	**ExampleInputStream**
flink.inputstream.initpos	**LATEST**
AWS_REGION	**us-west-2**

6. UNDER属性，选择创建组. 适用于Group ID (组 ID)输入**ProducerConfigProperties**.
7. 输入以下应用程序属性和值：

密钥	值
OUTPUT_STREAM	**ExampleOutputStream**
AWS_REGION	**us-west-2**
AggregationEnabled	**false**

8. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
9. 适用于CloudWatch 记录，选择启用”复选框。
10. 选择 Update（更新）。

Note

当你选择启用 CloudWatch 日志组和日Kinesis Data Analytics 和日志组日志组和日志志组日志组和日志志组日志 这些资源的名称如下所示：

- 日志组：`/aws/kinesis-analytics/MyApplication`
- 日志流：`kinesis-analytics-log-stream`

该日志流用于监控应用程序。这与应用程序用于发送结果的日志流不同。

运行应用程序

通过运行应用程序，打开 Apache Flink 控制面板，然后选择所需的 Flink 作业，可以查看 Flink 作业图。

你可以在上查看 Kinesis Data Analytics 指标 CloudWatch 控制台来验证应用程序是否正常工作。

您也可以在数据流的 Kinesis Data Streams 控制台中查看增强的扇出选项卡，用于您的消费者的姓名 (my-flink-efo-consumer)。

清理 Amazon 资源

本节包含清理过程Amazon在 efo 窗口教程中创建的资源。

本主题包含下列部分：

- 删除 Kinesis Data Analytics 应用程序 (p. 155)
- 删除 Kinesis Data Streams (p. 155)
- 删除您的 Amazon S3 对象和存储桶 (p. 155)
- 删除资源 (p. 155)
- 删除日期和时间 CloudWatch 资源 (p. 155)

删除 Kinesis Data Analytics 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 在应用程序的页面中，选择 Delete (删除)，然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream.
3. 在ExampleInputStream页面上，选择删除 Kinesis Streams然后确认删除。
4. 在Kinesis Streams页面上，选择ExampleOutputStream，选择操作，选择Delete，然后确认删除。

删除您的 Amazon S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择ka-app-code-***<username>***存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-***<your-region>***政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏中，选择 Roles（角色）。
7. 选择kinesines-anticticMyApplication-***<your-region>***角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除日期和时间 CloudWatch 资源

1. 打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/.
2. 在导航栏中，选择 Logs (日志)。
3. 选择/aws/kinesis-analytictics/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

示例：写入 Kinesis Data Firehose

在本练习中，您创建一个 Kinesis Data Analytics 应用程序，它将一个 Kinesis 数据流作为源，并将一个 Kinesis Data Firehose 传输流作为接收器。通过使用接收器，您可以验证 Amazon S3 存储桶中的应用程序输出。

> Note
>
> 要为本练习设置所需的先决条件，请先完成入门 (DataStreamAPI) (p. 76)练习。

本节包含以下步骤：

创建相关资源

在为本练习创建Kinesis Data Analytics for Apache For Apache Flink 应用程序之前，请创建以下相关资源：

- Kinesis Data Streams (`ExampleInputStream`)
- 应用程序将输出写入到的 Kinesis Data Firehose 传输流 (`ExampleDeliveryStream`)。
- Amazon S3 存储桶 (`ka-app-code-<username>`)

您可以使用控制台创建 Kinesis 流、Amazon S3 存储桶和 Kinesis Data Firehose 传输流。有关创建这些资源的说明，请参阅以下主题：

- 创建和更新数据流中的Amazon Kinesis Data Streams 开发者指南. 将数据流命名为 **`ExampleInputStream`**。
- 创建 Amazon Kinesis Data Firehose 传输流中的Amazon Kinesis Data Firehose. 将传输流命名为 **`ExampleDeliveryStream`**。在创建 Kinesis Data Firehose 传输流时，还要创建传输流的S3 目标和IAM 角色.
- 如何创建 S3 存储桶？中的Amazon Storage Service. 附加您的登录名，以便为 Amazon S3 存储桶指定全局唯一的名称，例如**`ka-app-code-<username>`**.

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入流，以供应用程序处理。

> **Note**
>
> 此部分需要 Amazon SDK for Python (Boto)。

1. 使用以下内容创建名为 `stock.py` 的文件：

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}
```

```
def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

2. 运行 `stock.py` 脚本：

```
$ python stock.py
```

在完成本教程的其余部分时，请将脚本保持运行状态。

下载并检查 Apache Flink 流式处理 Java 代码

该示例的 Java 应用程序代码可从 GitHub. 要下载应用程序代码，请执行以下操作：

1. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

2. 导航到 `amazon-kinesis-data-analytics-java-examples/FirehoseSink`目录。

应用程序代码位于 `FirehoseSinkStreamingJob.java` 文件中。请注意有关应用程序代码的以下信息：

- 应用程序使用 Kinesis 源从源流中进行读取。以下代码段创建了 Kinesis source：

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- 应用程序使用 Kinesis Data Firehose 接收器将数据写入到传输流。以下代码段创建了 Kinesis Data Firehose 接收器：

```
FlinkKinesisFirehoseProducer<String> sink = new
 FlinkKinesisFirehoseProducer<>(outputDeliveryStreamName, new SimpleStringSchema(),
 outputProperties);
```

编译应用程序代码

要编译应用程序，请执行以下操作：

1. 如果还没有 Java 和 Maven，请安装它们。有关更多信息，请参阅入门 (DataStreamAPI) (p. 76)教程中的先决条件 (p. 76)。
2. 要将 Kinesis 连接器用于以下应用程序，您需要下载、构建并安装 Apache Maven。有关更多信息，请参阅the section called “将 Apache Flink Kinesis Streams 连接器与早期 Apache Flink 版本一起使用” (p. 258)。
3. 使用以下命令编译应用程序：

```
mvn package -Dflink.version=1.13.2
```

Note

提供的源代码依赖于 Java 11 中的库。

编译应用程序将创建应用程序 JAR 文件 (`target/aws-kinesis-analytics-java-apps-1.0.jar`)。

上传 Apache Flink 流式处理 Java 代码

在本节中，您将应用程序代码上传到在一节中创建的 Amazon S3 存储桶创建相关资源 (p. 156)部分。

上传应用程序代码

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 在控制台中，选择ka-app-code-***<username>***存储桶，然后选择上传.
3. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `java-getting-started-1.0.jar` 文件。
4. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在存储Amazon S3 中，应用程序可以在其中访问代码。

创建和运行 Kinesis Data Analytics 应用程序

您可以使用控制台或使用 Kinesis Data Analytics 应用程序创建并运行 Kinesis 数据分析应用程序Amazon CLI.

Note

当您使用控制台创建应用程序时，Amazon Identity and Access Management(IAM) 和亚马逊 CloudWatch 日志资源是为您创建的。当您使用 Amazon CLI 创建应用程序时，您可以单独创建这些资源。

主题

- 创建并运行应用程序（控制台） (p. 158)
- 创建并运行应用程序 (Amazon CLI) (p. 161)

创建并运行应用程序（控制台）

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用程序.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于描述，输入 **`My java test app`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。

Note

Apache Flink Kinesis Data Analytics 使用 Apache Flink 版本 1.13.2。

- 将版本下拉菜单保留为Apache Flink 1.13.2（推荐版本）.

4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
5. 选择 Create application（创建应用程序）。

Note

在使用控制台创建应用程序时，您可以选择为您的应用程序创建 IAM 角色和策略。应用程序使用该角色和策略访问其相关资源。这些 IAM 资源是使用您的应用程序名称和区域命名的，如下所示：

- 策略：kinesis-analytics-service-*MyApplication*-*us-west-2*
- 角色：kinesis-analytics-*MyApplication*-*us-west-2*

编辑 IAM 策略

编辑 IAM 策略以添加访问 Kinesis Data Kinesis Data Firehose 流的权限。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择策略。选择控制台在上一部分中为您创建的 **kinesis-analytics-service-MyApplication-us-west-2** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。
4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*012345678901*) 的所有实例替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/java-getting-started-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
           ],
            "Resource": [
```

```
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteDeliveryStream",
            "Effect": "Allow",
            "Action": "firehose:*",
            "Resource": "arn:aws:firehose:us-west-2:012345678901:deliverystream/
ExampleDeliveryStream"
        }
    ]
}
```

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**ka-app-code-*<username>***.
   - 适用于Amazon S3 对象的路径输入**java-getting-started-1.0.jar**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
4. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
5. 适用于CloudWatch 记录，选择启用”复选框。
6. 选择 Update (更新) 。

   Note

   当你选择启用 CloudWatch 日志组和日Kinesis Data Analytics 和日志组日志组和日志志组日志组和日志志组日志 这些资源的名称如下所示：

   - 日志组：`/aws/kinesis-analytics/MyApplication`
   - 日志流：`kinesis-analytics-log-stream`

运行应用程序

通过运行应用程序，打开 Apache Flink 控制面板，然后选择所需的 Flink 作业，可以查看 Flink 作业图。

停止应用程序

在存储库的MyApplication页面上，选择停止. 确认该操作。

更新应用程序

使用控制台，您可以更新应用程序设置，例如应用程序属性、监控设置，或应用程序 JAR 文件的位置和文件名。

在存储库的MyApplication页面上，选择配置. 更新应用程序设置，然后选择更新。

Note

要在控制台上更新应用程序的代码，您必须更改 JAR 的对象名称，使用不同的 S3 存储桶，或使用the section called “更新应用程序代码” (p. 164)一节中所述的 Amazon CLI。如果文件名或存储桶未更改，则在您选择时不会重新加载应用程序代码更新在配置页.

创建并运行应用程序 (Amazon CLI)

在本部分中，您将使用Amazon CLI创建和运行 Kinesis Data Analytics 应用程序。

创建权限策略

首先，使用两个语句创建权限策略：一个语句授予对源流执行 `read` 操作的权限，另一个语句授予对接收器流执行 `write` 操作的权限。然后，将策略附加到一个 IAM 角色（下一部分中将创建此角色）。因此，在 Kinesis Data Analts 代入该角色时，服务具有必要的权限从源流进行读取和写入接收器流。

使用以下代码创建 `KAReadSourceStreamWriteSinkStream` 权限策略。Replace*###*将用于创建 Amazon S3 存储 将 Amazon 资源名称 (ARN) 中的账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream"
        },
        {
            "Sid": "WriteDeliveryStream",
            "Effect": "Allow",
            "Action": "firehose:*",
            "Resource": "arn:aws:firehose:us-west-2:012345678901:deliverystream/
ExampleDeliveryStream"
        }
    ]
}
```

适用于 step-by-step 创建权限策略的说明，请参阅教程：创建和附加您的第一个客户托管策略中的IAM 用户指南.

Note

要访问其他亚马逊服务，您可以使用Amazon SDK for Java. Kinesis Data Analts 会自动将开发工具包所需的凭证设置为与您的应用程序关联的服务执行 IAM 角色的凭证。无需执行其他步骤。

创建 IAM 角色

在本节中，您将创建一个 IAM 角色，Kinesis Data Analytics 应用程序可以代入此角色来读取源流和写入接收器流。

如果没有权限，Kinesis Data Analytics 将无法访问流。您通过 IAM 角色授予这些权限。每个 IAM 角色附加了两种策略。信任策略为授予 Kinesis Data Analytics 的权限，以担任该角色。权限策略确定在担任该角色后 Kinesis Data Analytics 可以执行的操作。

您将在上一部分中创建的权限策略附加到此角色。

创建 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles (角色) 和 Create Role (创建角色)。
3. UNDER选择受信任实体的类型，选择Amazon服务. 在 Choose the service that will use this role (选择将使用此角色的服务) 下，选择 Kinesis。在选择您的使用案例下，选择 Kinesis Analytics。

   选择 Next:。Permissions (下一步：权限)。
4. 在存储库的附加权限策略页面上，选择后续：审核。在创建角色后，您可以附加权限策略。
5. 在存储库的创建角色页面上，输入**`KA-stream-rw-role`**(对于)Role name (角色名称). 选择Create role（创建角色）。

   现在，您已经创建了一个名为的新 IAM 角色`KA-stream-rw-role`. 接下来，您更新角色的信任和权限策略。
6. 将权限策略附加到角色。

   Note

   在本练习中，Kinesis Data Analts Analts 代入此角色，以便同时从 Kinesis 数据流（源）读取数据和将输出写入另一个 Kinesis 数据流（源）。因此，您附加在上一步（the section called “创建权限策略” (p. 161)）中创建的策略。

   a. 在 Summary (摘要) 页上，选择 Permissions (权限) 选项卡。
   b. 选择附加策略。
   c. 在搜索框中，输入 **`KAReadSourceStreamWriteSinkStream`**（您在上一部分中创建的策略）。
   d. 选择KAReadSourceStreamWriteSinkStream策略，然后选择附加策略.

现在，您已创建应用程序用于访问资源的服务执行角色。记下新角色的 ARN。

适用于 step-by-step 有关创建角色的说明，请参阅创建 IAM 角色（控制台）中的IAM 用户指南.

创建 Kinesis Data Analytics 应用程序

1. 将以下 JSON 代码保存到名为 `create_request.json` 的文件中。将示例角色 ARN 替换为您之前创建的角色的 ARN。将存储桶 ARN 后缀替换为在the section called “创建相关资源” (p. 156)一节中选择的后缀 (`ka-app-code-<username>`)。将服务执行角色中的示例账户 ID (`012345678901`) 替换为您的账户 ID。

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_13",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/KA-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "java-getting-started-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        }
      }
    }
}
```

2. 使用上述请求执行 `CreateApplication` 操作来创建应用程序：

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

应用程序现已创建。您在下一步中启动应用程序。

启动应用程序

在本节中，您使用 `StartApplication` 操作来启动应用程序。

启动应用程序

1. 将以下 JSON 代码保存到名为 `start_request.json` 的文件中。

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. 使用上述请求执行 `StartApplication` 操作来启动应用程序：

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

应用程序正在运行。您可以在亚马逊上查看 Kinesis Data Analytics 指标 CloudWatch 控制台，以验证应用程序是否正常工作。

停止应用程序

在本节中，您使用 `StopApplication` 操作来停止应用程序。

停止应用程序

1. 将以下 JSON 代码保存到名为 `stop_request.json` 的文件中。

```
{
    "ApplicationName": "test"
}
```

2. 使用下面的请求执行 `StopApplication` 操作来停止应用程序：

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

应用程序现已停止。

添加 CloudWatch 日志记录选项

您可以使用Amazon CLI添加亚马逊 CloudWatch 将日志流发送到应用程序中。有关使用的信息 CloudWatch 使用应用程序的日志，请参阅the section called “设置日志记录” (p. 205).

更新应用程序代码

在您需要使用新版本的代码包更新应用程序代码时，您可以使用 `UpdateApplication` Amazon CLI 操作。

使用Amazon CLI，从 Amazon S3 存储桶中删除以前的代码包，上传新版本，然后调用调用新的版本并调用`UpdateApplication`，指定相同的 Amazon S3 存储桶桶和对象名称。

以下示例 `UpdateApplication` 操作请求重新加载应用程序代码并重新启动应用程序。将 `CurrentApplicationVersionId` 更新为当前的应用程序版本。您可以使用 `ListApplications` 或 `DescribeApplication` 操作检查当前的应用程序版本。将存储桶名称后缀 (<*username*>) 更新为在the section called “创建相关资源” (p. 156)一节中选择的后缀。

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "java-getting-started-1.0.jar"
                }
            }
        }
    }
}
```

清理 Amazon 资源

本节包含清理在入门教程中创建的 Amazon 资源的过程。

本主题包含下列部分：

删除 Kinesis Data Analytics 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 选择 Configure（配置）。
4. 在 Snapshots (快照) 部分中，选择 Disable (禁用)，然后选择 Update (更新)。
5. 在应用程序的页面中，选择 Delete (删除)，然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream.
3. 在ExampleInputStream页面上，选择删除 Kinesis Streams然后确认删除。

删除 Kinesis Data Firehose 传输流

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Firehose 面板中，选择ExampleDeliveryStream.
3. 在ExampleDeliveryStream页面上，选择删除 Delivery然后确认删除。

删除您的 Amazon S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择ka-app-code-***<username>***存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。
4. 如果您为 Kinesis Data Firehose 传输流目标创建了 Amazon S3 存储桶，则也会删除该存储桶。

删除资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-***<your-region>***政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 如果您为 Kinesis Data Firehose 传输流创建了新策略，则也会删除该策略。
7. 在导航栏中，选择 Roles（角色）。
8. 选择kinesines-anticticMyApplication-***<your-region>***角色。
9. 选择 Delete role (删除角色)，然后确认删除。
10. 如果您为 Kinesis Data Firehose 传输流创建了新角色，则也会删除该角色。

删除日期和时间 CloudWatch 资源

1. 打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/.
2. 在导航栏中，选择 Logs (日志)。
3. 选择/aws/kinesis-analytictics/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

示例：从不同账户的 Kinesis 流中读取

该示例说明了如何创建 Amazon Kinesis Data Analytics 应用程序，以从不同账户的 Kinesis 流中读取数据。在本示例中，您将一个账户用于源 Kinesis 流，并将第二个账户用于 Kinesis Data Analts 应用程序和接收器 Kinesis 流。

本主题包含下列部分：

先决条件

- 在本教程中，您将修改开始使用示例以从不同账户的 Kinesis 流中读取数据。在继续之前，请完成入门 (DataStreamAPI) (p. 76)教程。
- 您需要使用两个 Amazon 账户以完成本教程：一个账户用于源流，另一个账户用于应用程序和接收器流。将您用于入门教程的 Amazon 账户用于应用程序和接收器流。将一个不同的 Amazon 账户用于源流。

设置

您将使用命名的配置文件访问两个 Amazon 账户。修改您的Amazon凭证和配置文件以包含两个配置文件，其中包含两个账户的区域和连接信息。

以下示例凭证文件包含两个命名的配置文件：`ka-source-stream-account-profile` 和 `ka-sink-stream-account-profile`。将您用于入门教程的账户作为接收器流账户。

```
[ka-source-stream-account-profile]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

[ka-sink-stream-account-profile]
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
aws_secret_access_key=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
```

以下示例配置文件包含具有区域和输出格式信息的相同命名配置文件。

```
[profile ka-source-stream-account-profile]
region=us-west-2
output=json

[profile ka-sink-stream-account-profile]
region=us-west-2
output=json
```

> **Note**
>
> 本教程不使用 `ka-sink-stream-account-profile`。它是作为如何使用配置文件访问两个不同 Amazon 账户的示例提供的。

有关将命名配置文件与Amazon CLI，请参阅命名配置文件中的Amazon Command Line Interface文档中）。

创建源 Kinesis 流

在本节中，您在源账户中创建 Kinesis 流。

输入以下命令以创建 Kinesis 流，应用程序将该流用于输入。请注意，`--profile` 参数指定要使用的账户配置文件。

```
$ aws kinesis create-stream \
--stream-name SourceAccountExampleInputStream \
--shard-count 1 \
--profile ka-source-stream-account-profile
```

创建和更新 IAM 角色和策略

允许跨对象访问Amazon账户中，您可以在源账户中创建 IAM 角色和策略。然后，您在接收器账户中修改 IAM 策略。有关创建 IAM 角色和策略的信息，请参阅中的以下主题Amazon Identity and Access Management用户指南：

- 创建 IAM 角色
- 创建 IAM 策略

接收器账户角色和策略

1. 编辑入门教程中的 `kinesis-analytics-service-MyApplication-us-west-2` 策略。该策略允许担任源账户中的角色，以便读取源流。

   Note

   当您使用控制台创建应用程序时，控制台将创建一个名为`kinesis-analytics-service-<application name>-<application region>`，还有一个名为`kinesis-analytics-<application name>-<application region>`.

   将下面突出显示的部分添加到策略中。将示例账户 ID (*SOURCE01234567*) 替换为将用于源流的账户的 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AssumeRoleInSourceAccount",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::SOURCE01234567:role/KA-Source-Stream-Role"
        },
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
```

```
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:SINK012345678:log-group:*"
            ]
        },
        {
            "Sid": "ListCloudwatchLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:SINK012345678:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutCloudwatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:SINK012345678:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        }
    ]
}
```

2. 打开 kinesis-analytics-MyApplication-us-west-2 角色，并记下其 Amazon 资源名称 (ARN)。您需要在下一节中使用该名称。角色 ARN 如下所示。

```
arn:aws:iam::SINK012345678:role/service-role/kinesis-analytics-MyApplication-us-west-2
```

源账户角色和策略

1. 在名为 KA-Source-Stream-Policy 的源账户中创建一个策略。将以下 JSON 用于该策略。将示例账号替换为源账户的账号。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetRecords",
                "kinesis:GetShardIterator",
                "kinesis:ListShards"
            ],
            "Resource":
               "arn:aws:kinesis:us-west-2:SOURCE123456784:stream/
SourceAccountExampleInputStream"
        }
    ]
}
```

2. 在名为 KA-Source-Stream-Role 的源账户中创建一个角色。执行以下操作以使用 Kinesis Analytics 使用案例创建角色：

1. 在 IAM 管理控制台中，选择创建角色.
2. 在存储库的创建角色页面上，选择Amazon服务. 在服务列表中，选择 Kinesis。
3. 在 Select your use case (选择使用案例) 部分中，选择 Kinesis Analytics。
4. 选择 Next:。Permissions (下一步：权限)。
5. 添加您在上一步中创建的 `KA-Source-Stream-Policy` 权限策略。选择 Next:Tags (下一步: 标签)。
6. 选择 Next:。审核。
7. 将角色命名为 `KA-Source-Stream-Role`。应用程序将使用该角色以访问源流。

3. 将接收器账户中的 `kinesis-analytics-MyApplication-us-west-2` ARN 添加到源账户中的 `KA-Source-Stream-Role` 角色的信任关系中：

   1. 打开`KA-Source-Stream-Role`在 IAM 控制台中。
   2. 选择 Trust Relationships 选项卡。
   3. 选择 Edit trust relationship (编辑信任关系)。
   4. 将以下代码用于信任关系。将示例账户 ID (*SINK012345678*) 替换为接收器账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "AWS": "arn:aws:iam::SINK012345678:role/service-role/kinesis-analytics-
MyApplication-us-west-2"
        },
        "Action": "sts:AssumeRole"
      }
    ]
}
```

更新 Python 脚本

在本节中，您更新生成示例数据的 Python 脚本以使用源账户配置文件。

使用以下突出显示的更改更新 `stock.py` 脚本。

```
import json
import boto3
import random
import datetime
import os

os.environ['AWS_PROFILE'] ='ka-source-stream-account-profile'
os.environ['AWS_DEFAULT_REGION'] = 'us-west-2'


kinesis = boto3.client('kinesis')
def getReferrer():
    data = {}
    now = datetime.datetime.now()
    str_now = now.isoformat()
    data['EVENT_TIME'] = str_now
    data['ticker'] = random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV'])
    price = random.random() * 100
    data['price'] = round(price, 2)
    return data
```

```
while True:
        data = json.dumps(getReferrer())
        print(data)
        kinesis.put_record(
                StreamName="SourceAccountExampleInputStream",
                Data=data,
                PartitionKey="partitionkey")
```

更新 Java 应用程序

在本节中，您更新 Java 应用程序代码，以便从源流中读取时担任源账户角色。

对 BasicStreamingJob.java 文件进行以下更改。将示例源账号 (*SOURCE01234567*) 替换为您的源账号。

```
package com.amazonaws.services.kinesisanalytics;

import com.amazonaws.services.kinesisanalytics.runtime.KinesisAnalyticsRuntime;
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisProducer;
import org.apache.flink.streaming.connectors.kinesis.config.ConsumerConfigConstants;
import org.apache.flink.streaming.connectors.kinesis.config.AWSConfigConstants;

import java.io.IOException;
import java.util.Map;
import java.util.Properties;

 /**
 * A basic Kinesis Data Analytics for Java application with Kinesis data streams
 * as source and sink.
 */
public class BasicStreamingJob {
    private static final String region = "us-west-2";
    private static final String inputStreamName = "SourceAccountExampleInputStream";
    private static final String outputStreamName = ExampleOutputStream;
    private static final String roleArn = "arn:aws:iam::SOURCE01234567:role/KA-Source-
Stream-Role";
    private static final String roleSessionName = "ksassumedrolesession";

    private static DataStream<String>
 createSourceFromStaticConfig(StreamExecutionEnvironment env) {
        Properties inputProperties = new Properties();
        inputProperties.setProperty(AWSConfigConstants.AWS_CREDENTIALS_PROVIDER,
 "ASSUME_ROLE");
        inputProperties.setProperty(AWSConfigConstants.AWS_ROLE_ARN, roleArn);
        inputProperties.setProperty(AWSConfigConstants.AWS_ROLE_SESSION_NAME,
 roleSessionName);
        inputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
        inputProperties.setProperty(ConsumerConfigConstants.STREAM_INITIAL_POSITION,
 "LATEST");

        return env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new
 SimpleStringSchema(), inputProperties));
    }

    private static FlinkKinesisProducer<String> createSinkFromStaticConfig() {
        Properties outputProperties = new Properties();
        outputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
        outputProperties.setProperty("AggregationEnabled", "false");
```

```
        FlinkKinesisProducer<String> sink = new FlinkKinesisProducer<>(new
 SimpleStringSchema(), outputProperties);
        sink.setDefaultStream(outputStreamName);
        sink.setDefaultPartition("0");
        return sink;
    }

    public static void main(String[] args) throws Exception {
        // set up the streaming execution environment
        final StreamExecutionEnvironment env =
 StreamExecutionEnvironment.getExecutionEnvironment();

        DataStream<String> input = createSourceFromStaticConfig(env);

        input.addSink(createSinkFromStaticConfig());

        env.execute("Flink Streaming Java API Skeleton");
    }
}
```

构建、上传和运行应用程序

执行以下操作以更新和运行应用程序：

1. 在具有 `pom.xml` 文件的目录中运行以下命令，以再次构建应用程序。

   ```
   mvn package -Dflink.version=1.13.2
   ```

2. 从您的亚马逊简单存储服务 (Amazon S3) 存储桶中删除以前的 JAR 文件，然后上传新的`aws-kinesis-analytics-java-apps-1.0.jar`文件到 S3 存储桶。
3. 在 Kinesis Data Analytics 控制台的应用程序页面中，选择配置、更新重新加载应用程序 JAR 文件。
4. 运行 `stock.py` 脚本以将数据发送到源流。

   ```
   python stock.py
   ```

现在，应用程序从另一个账户的 Kinesis 流中读取数据。

您可以检查 `ExampleOutputStream` 流的 `PutRecords.Bytes` 指标，以验证应用程序是否正常工作。如果在输出流中具有活动，则应用程序正常工作。

教程：将自定义信任库与 Amazon MSK 结合使用

以下教程演示如何安全地连接（传输中的加密）到使用由自定义、私有甚至自托管的证书颁发机构 (CA) 颁发的服务器证书的 Kafka 集群。

为了通过 TLS 安全地将任何 Kafka 客户端连接到 Kafka 集群，Kafka 客户端（比如 Flink 应用程序示例）必须信任由 Kafka 集群的服务器证书（从颁发证书到根级 CA）提供的完整信任链。作为自定义信任库的示例，我们将使用启用了双向 TLS (MTLS) 身份验证的 Amazon MSK 集群。这意味着 MSK 群集节点使用由 AmazonCertificate Manager 私有证书颁发机构 (ACM 私有 CA) 对您的账户和区域是私有的，因此不被执行 Flink 应用程序的 Java 虚拟机 (JVM) 的默认信任库信任。

Note

- 一个密钥库用于存储应用程序应向服务器或客户端提供以进行验证的私钥和身份证书。
- 一个信托库用于存储来自认证机构 (CA) 的证书，用于验证服务器在 SSL 连接中提供的证书。

您还可以使用本教程中的技术在 Kinesis Data Analytics 应用程序与其他 Apache Kafka 源之间进行交互，例如：

- 托管在中的自定义 Apache Kafka 集群Amazon(Amazon EC2要么Amazon EKS)
- 一个Conflient Kafka集群托管在Amazon
- 通过访问的本地 Kafka 集群Amazon Direct ConnectVPN 节点

您的应用程序将使用自定义消费者 (`CustomFlinkKafkaConsumer`) 它会覆盖`open`方法来加载自定义信任库。这使得信任库在应用程序重新启动或替换线程后可供应用程序使用。

使用以下代码检索和存储自定义信任库，从`CustomFlinkKafkaConsumer.java`file:

```
@Override
public void open(Configuration configuration) throws Exception {
    // write truststore to /tmp
    // NOTE: make sure that truststore is in JKS format for KDA/Flink. See README for
 details
    dropFile("/tmp");

    super.open(configuration);
}

private void dropFile(String destFolder) throws Exception
{
    ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
    InputStream input = classLoader.getResourceAsStream("kafka.client.truststore.jks");
    byte[] buffer = new byte[input.available()];
    input.read(buffer);
        File destDir = new File(destFolder);
        File targetFile = new File(destDir, "kafka.client.truststore.jks");
        OutputStream outStream = new FileOutputStream(targetFile);
        outStream.write(buffer);
        outStream.flush();
}
```

> **Note**
>
> Apache Flink 要求信任库进入.

> **Note**
>
> 要为本练习设置所需的先决条件，请先完成入门 (DataStreamAPI) (p. 76)练习。

本教程包含以下部分：

使用亚马逊 MSK 集群创建 VPC

要创建示例 VPC 和 Amazon MSK 集群以从 Kinesis Data Analytics 应用程序中访问，请按照使用Amazon MSK 入门教程。

在完成本教程时，还执行以下操作：

- In第 5 步：创建主题，重复kafka-topics.sh --create命令以创建名为的目标主题AmazonKafkaTutorialTopicDestination：

```
bin/kafka-topics.sh --create --zookeeper ZooKeeperConnectionString --replication-factor 3
 --partitions 1 --topic AmazonKafkaTutorialTopicDestination
```

  Note

  如果kafka-topics.sh命令返回 aZooKeeperClientTimeoutException，验证 Kafka 集群的安全组是否有入站规则，允许来自客户端实例私有 IP 地址的所有流量。

- 记录集群的引导服务器列表。您可以使用以下命令获取引导服务器列表（替换*ClusterArn*使用您的 MSK 集群的 ARN）：

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
    "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094"
}
```

- 在执行本教程和必备教程中的步骤时，请务必使用所选的Amazon代码、命令和控制台条目中的区域。

创建自定义信任库并将其应用到您的集群

在本节中，您将创建自定义证书颁发机构 (CA)，使用它生成自定义信任库，然后将其应用于您的 MSK 集群。

要创建和应用您的自定义信任库，请按照客户端身份验证中的教程AmazManaged Streaming for Apache Kafka.

创建应用程序代码

在本节中，您下载并编译应用程序 JAR 文件。

该示例的 Java 应用程序代码可从 GitHub. 要下载应用程序代码，请执行以下操作：

1. 如果尚未安装 Git 客户端，请安装它。有关更多信息，请参阅安装 Git。
2. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

3. 应用程序代码位于 amazon-kinesis-data-analytics-java-examples/CustomKeystore/KDAFlinkStreamingJob.java 和 CustomFlinkKafkaConsumer.java 文件中。您可以检查代码以熟悉适用于 Apache Flink 的 Kinesis Data Analytics for Apache Flink 应用程序代码的结构。
4. 使用命令行 Maven 工具或首选的开发环境以创建 JAR 文件。要使用命令行 Maven 工具编译 JAR 文件，请输入以下内容：

```
mvn package -Dflink.version=1.13.2
```

   如果构建成功，则会创建以下文件：

```
target/flink-app-1.0-SNAPSHOT.jar
```

Note

提供的源代码依赖于 Java 11 中的库。如果你使用的是开发环境，

上传 Apache Flink 流式处理 Java 代码

在本节中，您将应用程序代码上传到在一节中创建的 Amazon S3 存储桶入门 (DataStreamAPI) (p. 76)教程。

Note

如果您从入门教程中删除了 Amazon S3 存储桶，请按照the section called “上传 Apache Flink 流式处理 Java 代码” (p. 82)再次进行。

1. 在 Amazon S3 控制台中，选择ka-app-code-***<username>***存储桶，然后选择上传.
2. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `KafkaGettingStartedJob-1.0.jar` 文件。
3. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在存储Amazon S3 中，应用程序可以在其中访问代码。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Amazon Kinesis Data Analytics 控制面板创建分析应用程序.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 适用于运行时，选择Flink 版本 1.13.2.
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
5. 选择 Create application（ 创建应用程序 ）。

Note

在使用控制台创建适用于 Apache Flink 的 Kinesis Data Analts Flink 应用程序时，您可以选择为应用程序创建一个 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源是使用您的应用程序名称和区域命名的，如下所示：

- 策略：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
- 角色：`kinesis-analytics-`*`MyApplication`*`-`*`us-west-2`*

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**`ka-app-code-`*`<username>`***.
   - 适用于Amazon S3 对象的路径输入**`flink-app-1.0-SNAPSHOT.jar`**.

3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。

   Note

   在使用控制台指定应用程序资源（例如，日志或 VPC）时，控制台修改应用程序执行角色以授予访问这些资源的权限。

4. 在 Properties (属性) 下面，选择 Add Group (添加组)。使用以下属性创建一个名为 **KafkaSource** 的属性组：

密钥	值
topic	AmazonKafkaTutorialTopic
bootstrap.servers	############
security.protocol	SSL
ssl.truststore.location	/usr/usr/usr/security/security/security/security/securityts
ssl.truststore.password	changeit

   Note

   这些区域有：ssl.truststore.password默认证书为 “changeit” — 如果您使用默认证书，则不需要更改该值。

   再次选择 Add Group (添加组)。使用以下属性创建一个名为 **KafkaSink** 的属性组：

密钥	值
topic	AmazonKafkaTutorialTopicDestination
bootstrap.servers	############
security.protocol	SSL
ssl.truststore.location	/usr/usr/usr/security/security/security/security/securityts
ssl.truststore.password	changeit
交易.timeout.ms	1000

   应用程序代码读取上述应用程序属性，以配置用于与 VPC 和 Amazon MSK 集群交互的源和接收器。有关使用属性的更多信息，请参阅运行时属性 (p. 18)。

5. 在 Snapshots (快照) 下面，选择 Disable (禁用)。这样，就可以轻松更新应用程序，而无需加载无效的应用程序状态数据。
6. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
7. 适用于CloudWatch 记录，选择启用”复选框。
8. 在 Virtual Private Cloud (VPC) 部分中，选择要与应用程序关联的 VPC。选择与您的 VPC 关联的子网和安全组，您希望应用程序使用它们访问 VPC 资源。
9. 选择 Update（更新）。

Note

当你选择启用 CloudWatch 日志组和日Kinesis Data Analytics 和日志组日志组和日志志组日志组和日志志组日志 这些资源的名称如下所示：

- 日志组：`/aws/kinesis-analytics/MyApplication`
- 日志流：`kinesis-analytics-log-stream`

该日志流用于监控应用程序。

运行应用程序

通过运行应用程序，打开 Apache Flink 控制面板，然后选择所需的 Flink 作业，可以查看 Flink 作业图。

测试应用程序

在本节中，您将记录写入到源主题。应用程序从源主题中读取记录，并将其写入到目标主题中。您可以将记录写入到源主题以及从目标主题中读取记录，以验证应用程序是否正常工作。

要写入和读取主题中的记录，请按照中的步骤操作步骤 6：生成和使用数据中的使用Amazon MSK 入门教程。

要从目标主题中读取，请在到集群的第二个连接中使用目标主题名称，而不是源主题：

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --consumer.config
 client.properties --topic AmazonKafkaTutorialTopicDestination --from-beginning
```

如果在目标主题中没有任何记录，请参阅故障排除 (p. 316)主题中的无法访问 VPC 中的资源 (p. 319)一节。

Python 示例

以下示例演示如何使用 Python 和 Apache Flink Table API 创建应用程序。

主题

- 示例：在 Python 中创建翻滚窗口 (p. 176)
- 示例：在 Python 中创建滑动窗口 (p. 183)
- 示例：使用 Python 将流数据发送至 Amazon S3 (p. 190)

示例：在 Python 中创建翻滚窗口

在本练习中，您创建一个使用滚动窗口聚合数据的 Python Kinesis Data Analytics 应用程序。

Note

要为本练习设置所需的先决条件，请先完成入门 (Python) (p. 103)练习。

本主题包含下列部分：

- 创建相关资源 (p. 177)
- 将示例记录写入输入流 (p. 177)
- 下载并检查应用程序代码 (p. 178)
- 压缩并上传 Apache Flink 流式处理 Python 代码 (p. 179)
- 创建和运行 Kinesis Data Analytics 应用程序 (p. 179)

- 清理 Amazon 资源 (p. 182)

创建相关资源

在为本练习创建 Kinesis Data Analytics 应用程序之前，请创建以下相关资源：

- 两个 Kinesis 数据流 (`ExampleInputStream`和`ExampleOutputStream`)
- Amazon S3 存储桶 (`ka-app-code-<username>`)

您可以使用控制台创建 Kinesis 流和 Amazon S3 存储桶。有关创建这些资源的说明，请参阅以下主题：

- 创建和更新数据流中的Amazon Kinesis Data Streams 开发者指南. 将数据流命名为 **`ExampleInputStream`** 和 **`ExampleOutputStream`**。
- 如何创建 S3 存储桶？中的Amazon Storage Service. 附加您的登录名，以便为 Amazon S3 存储桶指定全局唯一的名称，例如**`ka-app-code-<username>`**.

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入流，以供应用程序处理。

Note

此部分需要 Amazon SDK for Python (Boto)。

Note

本节中的 Python 脚本使用Amazon CLI. 你必须配置你的Amazon CLI使用您的账户凭证和默认区域。配置您的Amazon CLI，输入以下内容：

```
aws configure
```

1. 使用以下内容创建名为 `stock.py` 的文件：

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")
```

```
if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

2. 运行 stock.py 脚本：

```
$ python stock.py
```

在完成本教程的其余部分时，请将脚本保持运行状态。

下载并检查应用程序代码

该示例的 Python 应用程序代码可从 GitHub. 要下载应用程序代码，请执行以下操作：

1. 如果尚未安装 Git 客户端，请安装它。有关更多信息，请参阅安装 Git。
2. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

3. 导航到 amazon-kinesis-data-analytics-java-examples/python/TumblingWindow目录。

应用程序代码位于 tumbling-windows.py 文件中。请注意有关应用程序代码的以下信息：

- 应用程序使用 Kinesis 表源从源流中进行读取。以下片段调用了create_table函数来创建 Kinesis 表源：

```
table_env.execute_sql(
        create_table(input_table_name, input_stream, input_region, stream_initpos)
    )
```

这些区域有：create_table函数使用 SQL 命令创建由流式处理源支持的表：

```
def create_table(table_name, stream_name, region, stream_initpos):
    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time TIMESTAMP(3),
                WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND

              )
              PARTITIONED BY (ticker)
              WITH (
                'connector' = 'kinesis',
                'stream' = '{1}',
                'aws.region' = '{2}',
                'scan.stream.initpos' = '{3}',
                'sink.partitioner-field-delimiter' = ';',
                'sink.producer.collection-max-count' = '100',
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(
        table_name, stream_name, region, stream_initpos
    )
```

- 应用程序使用Tumble运算符聚合指定的翻滚窗口内的记录，并将聚合记录作为表对象返回：

```
tumbling_window_table = (
        input_table.window(
```

```
            Tumble.over("10.seconds").on("event_time").alias("ten_second_window")
        )
        .group_by("ticker, ten_second_window")
        .select("ticker, price.sum as price, ten_second_window.end as event_time")
    )
```

- 该应用程序使用 Kinesis Flink 连接器，来自amazon-kinesis-sql-connector-flink-2.12/1.13.2.jar.

压缩并上传 Apache Flink 流式处理 Python 代码

在本节中，您将应用程序代码上传到在一节中创建的 Amazon S3 存储桶创建相关资源 (p. 177)部分。

1. 使用您首选的压缩应用程序来压缩streaming-file-sink.py和flink-sql-connector-kinesis_2.12-1.13.2.jar文件。命名存档名称myapp.zip.
2. 在 Amazon S3 控制台中，选择ka-app-code-***<username>***存储桶，然后选择上传.
3. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 myapp.zip 文件。
4. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在存储Amazon S3 中，应用程序可以在其中访问代码。

创建和运行 Kinesis Data Analytics 应用程序

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用程序.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **MyApplication**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。

     Note

     Kinesis Data Analytics 使用 Apache Flink 版本 1.132。
   - 将版本下拉菜单保留为Apache Flink 1.13.2 (推荐版本) .
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
5. 选择 Create application (创建应用程序) 。

   Note

   在使用控制台创建 Kinesis Data Analts 应用程序时，您可以选择为应用程序创建一个 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源是使用您的应用程序名称和区域命名的，如下所示：

   - 策略：kinesis-analytics-service-*MyApplication*-*us-west-2*
   - 角色：kinesis-analytics-*MyApplication*-*us-west-2*

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.

2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**ka-app-code-*<username>***.
   - 适用于Amazon S3 对象的路径输入**myapp.zip**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
4. UNDER属性，选择添加组. 适用于Group ID (组 ID)输入**consumer.config.0**.
5. 输入以下应用程序属性和值：

密钥	值
input.stream.name	**ExampleInputStream**
aws.region	**us-west-2**
flink.stream.initpos	**LATEST**

   选择Save（保存）。
6. UNDER属性，选择添加组。适用于Group ID (组 ID)输入**producer.config.0**.
7. 输入以下应用程序属性和值：

密钥	值
output.stream.name	**ExampleOutputStream**
aws.region	**us-west-2**
shard.count	**1**

8. UNDER属性，选择添加组。适用于Group ID (组 ID)输入**kinesis.analytics.flink.run.options**. 这个特殊的属性组告诉你的应用程序在哪里可以找到它的代码资源。有关更多信息，请参阅 指定代码文件 (p. 17)。
9. 输入以下应用程序属性和值：

密钥	值
python	**tumbling-windows.py**
jarfile	**flink-sql-connector-kinesis_2.12-1.13.2.jar**

10. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
11. 适用于CloudWatch 记录，选择启用”复选框。
12. 选择 Update（更新）。

Note

当你选择启用 CloudWatch 日志记录时，Kinesis Data Analytics 将为您创建日志组和日志流。这些资源的名称如下所示：

- 日志组：`/aws/kinesis-analytics/MyApplication`

- 日志流：`kinesis-analytics-log-stream`

该日志流用于监控应用程序。这与应用程序用于发送结果的日志流不同。

编辑 IAM 策略

编辑 IAM 策略添加访问数据流数据流的Kinesis 限限限。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择策略。选择控制台在上一部分中为您创建的 **`kinesis-analytics-service-MyApplication-us-west-2`** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。
4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/myapp.zip"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
```

```
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

运行应用程序

通过运行应用程序，打开 Apache Flink 控制面板，然后选择所需的 Flink 作业，可以查看 Flink 作业图。

你可以在上查看 Kinesis Data Analytics 指标 CloudWatch 控制台，以验证应用程序是否正常工作。

清理 Amazon 资源

本节包含清理在滚动窗口教程中创建的 Amazon 资源的过程。

本主题包含下列部分：

- 删除 Kinesis Data Analytics 应用程序 (p. 182)
- 删除 Kinesis Data Streams (p. 182)
- 删除您的 Amazon S3 对象和存储桶 (p. 182)
- 删除资源 (p. 182)
- 删除日期和时间 CloudWatch 资源 (p. 183)

删除 Kinesis Data Analytics 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 在应用程序的页面中，选择 Delete (删除)，然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream.
3. 在ExampleInputStream页面上，选择删除 Kinesis Streams然后确认删除。
4. 在Kinesis Streams页面上，选择ExampleOutputStream，选择操作，选择Delete，然后确认删除。

删除您的 Amazon S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择ka-app-code-***<username>***存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。

2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-*<your-region>*政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏中，选择 Roles（角色）。
7. 选择kinesines-anticticMyApplication-*<your-region>*角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除日期和时间 CloudWatch 资源

1. 打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/.
2. 在导航栏中，选择 Logs (日志)。
3. 选择/aws/kinesis-analytictics/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

示例：在 Python 中创建滑动窗口

在本练习中，您创建一个使用滑动窗口聚合数据的 Python Kinesis Data Analytics 应用程序。在 Flink 中，该功能状态。要禁用终止保护，请使用以下内容：

```
sink.producer.aggregation-enabled' = 'false'
```

> **Note**
>
> 要为本练习设置所需的先决条件，请先完成入门 (Python) (p. 103)练习。

本主题包含下列部分：

- 创建相关资源 (p. 183)
- 将示例记录写入输入流 (p. 184)
- 下载并检查应用程序代码 (p. 184)
- 压缩并上传 Apache Flink 流式处理 Python 代码 (p. 185)
- 创建和运行 Kinesis Data Analytics 应用程序 (p. 186)
- 清理 Amazon 资源 (p. 189)

创建相关资源

在为本练习创建 Kinesis Data Analytics 应用程序之前，请创建以下相关资源：

- 两个 Kinesis 数据流 (`ExampleInputStream`和`ExampleOutputStream`)
- Amazon S3 存储桶 (`ka-app-code-<username>`)

您可以使用控制台创建 Kinesis 流和 Amazon S3 存储桶。有关创建这些资源的说明，请参阅以下主题：

- 创建和更新数据流中的Amazon Kinesis Data Streams 开发者指南. 将数据流命名为 **`ExampleInputStream`** 和 **`ExampleOutputStream`**。
- 如何创建 S3 存储桶？中的Amazon Storage Service. 附加您的登录名，以便为 Amazon S3 存储桶指定全局唯一的名称，例如**`ka-app-code-<username>`**.

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入流，以供应用程序处理。

Note

此部分需要 Amazon SDK for Python (Boto)。

Note

本节中的 Python 脚本使用Amazon CLI. 你必须配置你的Amazon CLI使用您的账户凭证和默认区域。配置您的Amazon CLI，输入以下内容：

```
aws configure
```

1. 使用以下内容创建名为 stock.py 的文件：

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

2. 运行 stock.py 脚本：

```
$ python stock.py
```

在完成本教程的其余部分时，请将脚本保持运行状态。

下载并检查应用程序代码

该示例的 Python 应用程序代码可从 GitHub. 要下载应用程序代码，请执行以下操作：

1. 如果尚未安装 Git 客户端，请安装它。有关更多信息，请参阅安装 Git。
2. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/>amazon-kinesis-data-analytics-java-examples
```

3. 导航到 amazon-kinesis-data-analytics-java-examples/python/SlidingWindow目录。

应用程序代码位于 sliding-windows.py 文件中。请注意有关应用程序代码的以下信息：

- 应用程序使用 Kinesis 表源从源流中进行读取。以下片段调用了create_table函数来创建 Kinesis 表源：

```
table_env.execute_sql(
        create_table(input_table_name, input_stream, input_region, stream_initpos)
    )
```

这些区域有：create_table函数使用 SQL 命令创建由流式处理源支持的表：

```
def create_table(table_name, stream_name, region, stream_initpos):
    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time TIMESTAMP(3),
                WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND

              )
              PARTITIONED BY (ticker)
              WITH (
                'connector' = 'kinesis',
                'stream' = '{1}',
                'aws.region' = '{2}',
                'scan.stream.initpos' = '{3}',
                'sink.partitioner-field-delimiter' = ';',
                'sink.producer.collection-max-count' = '100',
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(
        table_name, stream_name, region, stream_initpos
    )
```

- 应用程序使用Slide运算符聚合指定滑动窗口内的记录，并将聚合记录作为表对象返回：

```
sliding_window_table = (
        input_table.window(
            Slide.over("10.seconds")
            .every("5.seconds")
            .on("event_time")
            .alias("ten_second_window")
        )
        .group_by("ticker, ten_second_window")
        .select("ticker, price.min as price, ten_second_window.end as event_time")
    )
```

- 该应用程序使用 Kinesis Flink 连接器，来自flink-sql-connector-kinesis_2.12文件。

压缩并上传 Apache Flink 流式处理 Python 代码

在本节中，您将应用程序代码上传到在一节中创建的 Amazon S3 存储桶创建相关资源 (p. 183)部分。

1. 使用您首选的压缩应用程序来压缩streaming-file-sink.py和flink-sql-connector-kinesis_2.12-1.13.2.jar文件。命名存档名称myapp.zip.

2. 在 Amazon S3 控制台中，选择ka-app-code-***<username>***存储桶，然后选择上传.
3. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `myapp.zip` 文件。
4. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在存储Amazon S3 中，应用程序可以在其中访问代码。

创建和运行 Kinesis Data Analytics 应用程序

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用程序.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。

     Note

     Kinesis Data Analytics 使用 Apache Flink 版本 1.132。
   - 将版本下拉菜单保留为Apache Flink 1.13.2（推荐版本）.
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
5. 选择 Create application（创建应用程序）。

   Note

   在使用控制台创建 Kinesis Data Analts 应用程序时，您可以选择为应用程序创建一个 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源是使用您的应用程序名称和区域命名的，如下所示：

   - 策略：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
   - 角色：`kinesis-analytics-`*`MyApplication`*`-`*`us-west-2`*

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**`ka-app-code-`*`<username>`***.
   - 适用于Amazon S3 对象的路径输入**`myapp.zip`**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
4. UNDER属性，选择添加组. 适用于Group ID (组 ID)输入**`consumer.config.0`**.
5. 输入以下应用程序属性和值：

密钥	值
`input.stream.name`	**`ExampleInputStream`**

密钥	值
aws.region	**us-west-2**
flink.stream.initpos	**LATEST**

选择Save（保存）。

6. UNDER属性，选择添加组。适用于Group ID (组 ID)输入**producer.config.0**.
7. 输入以下应用程序属性和值：

密钥	值
output.stream.name	**ExampleOutputStream**
aws.region	**us-west-2**
shard.count	**1**

8. UNDER属性，选择添加组。适用于Group ID (组 ID)输入**kinesis.analytics.flink.run.options**. 这个特殊的属性组告诉你的应用程序在哪里可以找到它的代码资源。有关更多信息，请参阅 指定代码文件 (p. 17)。
9. 输入以下应用程序属性和值：

密钥	值
python	**sliding-windows.py**
jarfile	**amazon-kinesis-connector-flink-2.0.0.jar**

10. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
11. 适用于CloudWatch 记录，选择启用”复选框。
12. 选择 Update（更新）。

Note

当你选择启用 CloudWatch 日志记录时，Kinesis Data Analytics 将为您创建日志组和日志流。这些资源的名称如下所示：

- 日志组：`/aws/kinesis-analytics/MyApplication`
- 日志流：`kinesis-analytics-log-stream`

该日志流用于监控应用程序。这与应用程序用于发送结果的日志流不同。

编辑 IAM 策略

编辑 IAM 策略添加访问数据流数据流的Kinesis 限限限。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择策略。选择控制台在上一部分中为您创建的 **kinesis-analytics-service-MyApplication-us-west-2** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。

4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/myapp.zip"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

运行应用程序

通过运行应用程序，打开 Apache Flink 控制面板，然后选择所需的 Flink 作业，可以查看 Flink 作业图。

你可以在上查看 Kinesis Data Analytics 指标 CloudWatch 控制台，以验证应用程序是否正常工作。

清理 Amazon 资源

本节包含清理在滑动窗口教程中创建的 Amazon 资源的过程。

本主题包含下列部分：

删除 Kinesis Data Analytics 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 在应用程序的页面中，选择 Delete (删除)，然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream.
3. 在ExampleInputStream页面上，选择删除 Kinesis Streams然后确认删除。
4. 在Kinesis Streams页面上，选择ExampleOutputStream，选择操作，选择Delete，然后确认删除。

删除您的 Amazon S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择ka-app-code-***<username>***存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-***<your-region>***政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏中，选择 Roles（角色）。
7. 选择kinesines-anticticMyApplication-***<your-region>***角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除日期和时间 CloudWatch 资源

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

2. 在导航栏中，选择 Logs (日志)。
3. 选择/aws/kinesis-analytictics/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

示例：使用 Python 将流数据发送至 Amazon S3

在本练习中，您将创建一个 Python Kinesis Data Analytics 应用程序，该应用程序将数据流式传输到亚马逊简单存储服务接收器。

> Note
>
> 要为本练习设置所需的先决条件，请先完成入门 (Python) (p. 103)练习。

本主题包含下列部分：

创建相关资源

在为本练习创建 Kinesis Data Analytics 应用程序之前，请创建以下相关资源：

- Kinesis Data Streams (`ExampleInputStream`)
- 用于存储应用程序代码和输出的 Amazon S3 存储段 (`ka-app-code-<username>`)

> Note
>
> 如果在 Kinesis Data Analytics 上启用了服务器端加密，适用于 Apache Flink 的 Kinesis Data Analytics 无法将数据写入到 Amazon S3。

您可以使用控制台创建 Kinesis 流和 Amazon S3 存储桶。有关创建这些资源的说明，请参阅以下主题：

- 创建和更新数据流中的Amazon Kinesis Data Streams 开发者指南. 将数据流命名为 **`ExampleInputStream`**。
- 如何创建 S3 存储桶？中的Amazon Storage Service. 附加您的登录名，以便为 Amazon S3 存储桶指定全局唯一的名称，例如**`ka-app-code-<username>`**.

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入流，以供应用程序处理。

> Note
>
> 此部分需要 Amazon SDK for Python (Boto)。

> Note
>
> 本节中的 Python 脚本使用Amazon CLI. 你必须配置你的Amazon CLI使用您的账户凭证和默认区域。配置您的Amazon CLI，输入以下内容：

```
aws configure
```

1. 使用以下内容创建名为 stock.py 的文件：

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

2. 运行 stock.py 脚本：

```
$ python stock.py
```

在完成本教程的其余部分时，请将脚本保持运行状态。

下载并检查应用程序代码

该示例的 Python 应用程序代码可从 GitHub. 要下载应用程序代码，请执行以下操作：

1. 如果尚未安装 Git 客户端，请安装它。有关更多信息，请参阅安装 Git。
2. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

3. 导航到 amazon-kinesis-data-analytics-java-examples/python/S3Sink目录。

应用程序代码位于 getting-started.py 文件中。请注意有关应用程序代码的以下信息：

- 应用程序使用 Kinesis 表源从源流中进行读取。以下片段调用了create_table函数来创建 Kinesis 表源：

```
table_env.execute_sql(
        create_table(input_table_name, input_stream, input_region, stream_initpos)
```

```
    )
```

这些区域有：`create_table`函数使用 SQL 命令创建由流式处理源支持的表：

```
def create_table(table_name, stream_name, region, stream_initpos):
    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time TIMESTAMP(3),
                WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND

              )
              PARTITIONED BY (ticker)
              WITH (
                'connector' = 'kinesis',
                'stream' = '{1}',
                'aws.region' = '{2}',
                'scan.stream.initpos' = '{3}',
                'sink.partitioner-field-delimiter' = ';',
                'sink.producer.collection-max-count' = '100',
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(
        table_name, stream_name, region, stream_initpos
    )
```

- 应用程序使用`filesystem`连接器，用于向 Amazon S3 存储桶发送记录：

```
def create_sink_table(table_name, bucket_name):
    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time TIMESTAMP(3),
                WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND

              )
              PARTITIONED BY (ticker)
              WITH (
                  'connector'='filesystem',
                  'path'='s3a://{1}/',
                  'format'='csv',
                  'sink.partition-commit.policy.kind'='success-file',
                  'sink.partition-commit.delay' = '1 min'
              ) """.format(
        table_name, bucket_name)
```

- 该应用程序使用 Kinesis Flink 连接器，来自amazon-kinesis-connector-flink-2.12.jar文件。

压缩并上传 Apache Flink 流式处理 Python 代码

在本节中，您将应用程序代码上传到在一节中创建的 Amazon S3 存储桶创建相关资源 (p. 190)部分。

1. 使用您首选的压缩应用程序来压缩`streaming-file-sink.py`和flink-sql-connector-kinesis_2.2.12-outparameter文件。命名存档名称`myapp.zip`.
2. 在 Amazon S3 控制台中，选择ka-app-code-***<username>***存储桶，然后选择上传.
3. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `myapp.zip` 文件。
4. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在存储Amazon S3 中，应用程序可以在其中访问代码。

创建和运行 Kinesis Data Analytics 应用程序

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用程序.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。

     Note

     Kinesis Data Analytics 使用 Apache Flink 版本 1.132。
   - 将版本下拉菜单保留为Apache Flink 1.13.2（推荐版本）.
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
5. 选择 Create application（创建应用程序）。

   Note

   在使用控制台创建 Kinesis Data Analts 应用程序时，您可以选择为应用程序创建一个 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源是使用您的应用程序名称和区域命名的，如下所示：

   - 策略：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
   - 角色：`kinesis-analytics-`*`MyApplication`*`-`*`us-west-2`*

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶，输入**`ka-app-code-`**`<username>`.
   - 适用于Amazon S3 对象的路径，输入**`myapp.zip`**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
4. UNDER属性，选择添加组. 适用于Group ID (组 ID)，输入**`consumer.config.0`**.
5. 输入以下应用程序属性和值：

密钥	值
`input.stream.name`	**`ExampleInputStream`**
`aws.region`	**`us-west-2`**
`flink.stream.initpos`	**`LATEST`**

选择Save（保存）。

6. UNDER属性，选择添加组。适用于Group ID (组 ID)，输入**kinesis.analytics.flink.run.options**. 这个特殊的属性组告诉你的应用程序在哪里可以找到它的代码资源。有关更多信息，请参阅 指定代码文件 (p. 17)。
7. 输入以下应用程序属性和值：

密钥	值
python	**streaming-file-sink.py**
jarfile	**S3Sink/lib/flink-sql-connector-kinesis_2.12-1.13.2.jar**

8. UNDER属性，选择添加组。适用于Group ID (组 ID)，输入**sink.config.0**. 这个特殊的属性组告诉你的应用程序在哪里可以找到它的代码资源。有关更多信息，请参阅 指定代码文件 (p. 17)。
9. 输入以下应用程序属性和值：(*bucket-name*替换为您的 Amazon S3 存储桶的实际名称。)

密钥	值
output.bucket.name	***bucket-name***

10. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
11. 适用于CloudWatch 记录，选择启用”复选框。
12. 选择 Update (更新) 。

Note

当你选择启用 CloudWatch 日志记录时，Kinesis Data Analytics 将为您创建日志组和日志流。这些资源的名称如下所示：

- 日志组：`/aws/kinesis-analytics/MyApplication`
- 日志流：`kinesis-analytics-log-stream`

该日志流用于监控应用程序。这与应用程序用于发送结果的日志流不同。

编辑 IAM 策略

编辑 IAM 策略添加访问数据流数据流的Kinesis 限限限。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择策略。选择控制台在上一部分中为您创建的 **kinesis-analytics-service-MyApplication-us-west-2** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。
4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
```

```
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/myapp.zip"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteObjects",
            "Effect": "Allow",
            "Action": [
                "s3:Abort*",
                "s3:DeleteObject*",
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-<username>",
                "arn:aws:s3:::ka-app-<username>/*"
            ]
        }
    ]
}
```

运行应用程序

通过运行应用程序，打开 Apache Flink 控制面板，然后选择所需的 Flink 作业，可以查看 Flink 作业图。

你可以在上查看 Kinesis Data Analytics 指标 CloudWatch 控制台，以验证应用程序是否正常工作。

清理 Amazon 资源

本节包含清理在滑动窗口教程中创建的 Amazon 资源的过程。

本主题包含下列部分：

删除 Kinesis Data Analytics 应用程序

1. 打开 Kinesis Data Analytics 控制台，网址为https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 在应用程序的页面中，选择 Delete (删除)，然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream.
3. 在ExampleInputStream页面上，选择删除 Kinesis Streams然后确认删除。

删除Amazon S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择ka-app-code-***<username>***存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-***<your-region>***政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏中，选择 Roles（角色）。
7. 选择kinesines-anticticMyApplication-***<your-region>***角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除日期和时间 CloudWatch 资源

1. 打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/.
2. 在导航栏中，选择 Logs (日志)。
3. 选择/aws/kinesis-analytictics/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

Amazon Kinesis Data Analytics 中的安全性

Amazon 十分重视云安全性。作为 Amazon 客户，您将会从专为满足大多数安全敏感型组织的要求而打造的数据中心和网络架构中受益。

安全性是 Amazon 和您的共同责任。责任共担模型将其描述为云的安全性和云中的安全性：

- 云的安全性 – Amazon负责保护在Amazon云中运行Amazon服务的基础设施。Amazon还向您提供可安全使用的服务。作为 Amazon 合规性计划的一部分，我们的安全措施的有效性定期由第三方审计员进行测试和验证。要了解适用于 Kinesis Data Analytics 的合规性计划，请参阅Amazon合规性计划范围内的服务.
- 云中的安全性 - 您的责任由您使用的 Amazon 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您组织的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 Kinesis Data Analytics 时应用责任共担模式。以下主题说明如何配置 Kinesis Data Analytics 以实现您的安全性和合规性目标。您还将了解如何使用其他 Amazon 服务，以帮助您监控和保护您的 Kinesis Data Analytics 资源。

主题

Amazon Kinesis Data Analytics 中的数据保护

您可以使用提供的工具保护您的数据Amazon. Kinesis Data Analytics 可以与支持加密数据的服务配合使用，包括 Kinesis Data Analytics、Kinesis Data Firehose 和 Amazon S3。

Kinesis Data Analytics 中的数据加密

静态加密

在使用适用于 Apache Flink 的 Kinesis Data Analytics 加密静态数据时，请注意以下事项：

- 您可以使用对传入的 Kinesis 数据流上的数据进行加密StartStreamEncryption. 有关更多信息，请参阅什么是 Kinesis 数据流的服务器端加密？。
- 输出数据可以使用Kinesis Data Firehose进行静态加密，以将数据存储在加密的 Amazon S3 存储桶中。您可以指定 Amazon S3 存储桶使用的加密密钥。有关更多信息，请参阅 。使用具有 KMS 托管密的服务器端加密 (SSE-KMS) 保护数据.
- Kinesis Data Analytics for Apache Flink 应用程序可以从任何流式传输源中读取，并写入到任何流式传输或数据库目标。确保源和目标对传输中的所有数据和静态数据进行加密。

- 您的应用程序的代码是静态加密的。
- 持久应用程序存储是静态加密的。
- 运行中的应用程序存储是静态加密的。

传输中加密

Kinesis Data Analytics 对所有传输中的数据进行加密。传输中加密对所有 Kinesis Data Analytics 应用程序都处于启用状态，无法禁用。

在以下场景中，Kinesis Data Analytics 对传输中的数据进行加密：

- 从 Kinesis Data Streams 到 Kinesis Data Analytics 传输中的数据。
- 在 Kinesis Data Analytics 的内部组件之间传输中的数据。
- 在 Kinesis Data Analytics 和 Kinesis Data Firehose 之间传输的数据。

密钥管理

Kinesis Data Analytics 中的数据加密使用服务托管的密钥。客户管理的密钥不受支持。

Amazon Kinesis Data Analytics for Apache Flink

Amazon Kinesis Data Analytics 需要适当权限，以从应用程序配置上指定的流式传输源中读取记录。Kinesis Data Analytics 还需要具有相应的权限，以将应用程序输出写入到您在应用程序配置上指定的接收器中。

> Note
>
> 您必须为应用程序创建一个权限策略和角色。如果你不创建这些Amazon Identity and Access Management(IAM) 资源，应用程序无法访问其数据源、数据目标和日志流。

您可以通过创建 Kinesis Data Analytics 可担任的 IAM 角色来授予这些权限。您授予此角色的权限确定了 Kinesis Data Analytics 在担任该角色时可以执行的操作。

> Note
>
> 如果要自己创建 IAM 角色，此部分中的信息是非常有用的。在 Kinesis Data Analytics 控制台中创建应用程序时，控制台可以为您创建 IAM 角色。对于控制台创建的 IAM 角色，控制台使用以下命名约定。
>
> ```
> kinesis-analytics-ApplicationName
> ```
>
> 在创建角色后，您可以在中查看该角色和附加的策略Amazon Identity and Access Management(IAM) 控制台。

每个 IAM 角色附加了两个策略。在信任策略中，您可以指定谁可以代入该角色。在权限策略（可以有一个或多个）中，您应当指定要向此角色授予的权限。以下部分说明了这些策略，您可以在创建 IAM 角色时使用这些策略。

主题

信任策略

要向 Kinesis Data Analts Analts 授予代入某个角色的权限以访问流式传输源或引用源，您可以将以下信任策略附加到 IAM 角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

权限策略

如果您正在创建 IAM 角色以允许 Kinesis Data Analytics 从应用程序的流式传输源读取，您必须授予相关读取操作的权限。流式传输源示例包括 Kinesis 数据流、Amazon Kinesis Data Firehose 传输流或Amazon Simple Storage Service (Amazon S3) 存储桶中的引用源。根据源，您可以附加以下权限策略。

读取Kinesis Data 流的权限策略

在以下示例中，将每个*#######*替换为您自己的信息。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadInputKinesis",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": [
                "arn:aws:kinesis:aws-region:aws-account-id:stream/inputStreamName"
            ]
        }
    ]
}
```

用于写入 Kinesis Data 流的权限策略

在以下示例中，将每个*#######*替换为您自己的信息。

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

```
                "kinesis:DescribeStream",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:aws-region:aws-account-id:stream/output-stream-name"
            ]
        }
    ]
}
```

写入到 Kinesis Data Firehose 传输流的权限策略

在以下示例中，将每个*#######*替换为您自己的信息。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "WriteOutputFirehose",
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": [
                "arn:aws:firehose:aws-region:aws-account-id:deliverystream/output-firehose-
name"
            ]
        }
    ]
}
```

用于从 Amazon S3 存储桶

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "*"
    }
  ]
}
```

用于写入到的权限策略 CloudWatch 日志流

有关添加权限以写入到 CloudWatch 日志流，请参阅将写入权限添加到 CloudWatch 日志流 (p. 209).

监控Amazon Kinesis Data Analytics

Kinesis Data Analytics 为您的应用程序提供监控功能。有关更多信息，请参阅 日志记录和监控 (p. 204)。

Amazon Kinesis Data Analytics 的合规性验证

作为多项计划的一部分，第三方审计员将评估 Amazon Kinesis Data Analytics for Apache Flink 的安全性和合规性Amazon合规性计划。其中包括 SOC、PCI、HIPAA 等。

列表Amazon特定合规性计划范围内的服务，请参阅。合规性计划范围内的 Amazon Web Services. 有关一般信息，请参阅Amazon合规性计划。

您可以使用 Amazon Artifact 下载第三方审计报告。有关更多信息，请参见下载 Amazon Artifact 中的报告。

您在使用 Kinesis Data Analts Flink 时的合规性责任由您的数据的敏感性、您公司的合规性目标以及适用的法律法规决定。如果您对 Kinesis Data Analytics for Apache Flink 的使用需遵守 HIPAA 或 PCI 等标准，Amazon提供资源，以帮助：

- 《安全性与合规性快速入门指南》– 这些部署指南讨论了架构注意事项，并提供了在Amazon上部署基于安全性和合规性的基准环境的步骤。
- 设计符合 HIPAA 安全性和合规性要求的架构白皮书 – 此白皮书介绍公司如何使用 Amazon 创建符合 HIPAA 标准的应用程序。
- Amazon 合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- Amazon Config – 此Amazon服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- Amazon Security Hub – 此 Amazon 服务提供了 Amazon 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。

FedRAMP

这些区域有：AmazonFedRAMP 合规性计划包括作为 FedRAMP 授权服务的 Kinesis Data Analytics。如果您是联邦客户或商业客户，则可以使用该服务来处理您的敏感型工作负载并将这些负载存储在Amazon GovCloud （美国）区域的授权范围，数据达到高影响级别以及数据达到中等级别的美国东部（弗吉尼亚北部）、美国东部（俄勒冈）、美国东部（俄勒冈）、美国东部（俄勒冈）、美国东部（俄勒冈）、美国东部（俄勒冈）、美国东部（俄勒冈）、美国东部（北部（俄勒冈）、美国东部（北部（

你可以请求访问Amazon通过 FedRAMP 项目管理办公室或您的 FedRAMP 安全套餐Amazon销售客户经理，或者，他们可以通过AmazonArtifactAmazonArtifact.

有关更多信息，请参阅 。FedRAMP.

Amazon Kinesis Data Analytics 中的弹性

Amazon全球基础设施围绕Amazon区域和可用区构建。Amazon区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 Amazon 区域和可用区的更多信息，请参阅 Amazon 全球基础设施。

除了AmazonKinesis Data Analytics for Apache Flink 提供了多种功能，以帮助支持您的数据恢复能力和备份需求。

灾难恢复

Kinesis Data Analts 在无服务器模式下运行，通过执行自动迁移来处理主机降级、可用区可用性以及其他基础设施相关问题。Kinesis Data Analytics 通过多种冗余机制实现这一目标。每个使用 Apache Flink 的

Kinesis Data Analytics 应用程序在单租户 Apache Flink 集群中运行。Apache Flink 集群使用 JobMananger 在高可用性模式下跨多个可用区使用 Zookeeper。Kinesis Data Analytics 使用亚马逊 EKS 部署 Apache Flink。Amazon EKS EKS 中为每个 Amazon EKs 存储桶Amazon跨可用区的区域。如果发生故障，Kinesis Data Analts 首先尝试使用应用程序的检查点（如果可用）在运行的 Apache Flink 集群中恢复应用程序。

Kinesis Data Analytics for Apache Flink 使用备份应用程序状态检查点和快照：

- 检查点是应用程序状态备份，Kinesis Data Analytics 定期自动创建这些备份并用于从故障中还原。
- 快照 是您手动创建的应用程序状态备份，可以从这些备份中进行还原。

有关检查点和快照的更多信息，请参阅容错能力 (p. 21)。

版本控制

存储的应用程序状态版本按如下方式进行版本控制：

- 该服务自动对检查点 进行版本控制。如果该服务使用检查点重新启动应用程序，则会使用最新的检查点。
- 保存点使用版本控制SnapshotNameParameterCreateApplicationSnapshotaction.

Kinesis Data Analytics 对在检查点和保存点中存储的数据进行加密。

Kinesis Data Analytics 中的基础设施安全性

作为一项托管服务，Amazon Kinesis Data Analytics 受到Amazon中描述的全局网络安全程序Amazon Web Services：安全过程概述白皮书。

你用Amazon发布的 API 调用通过网络访问 Kinesis Data Analytics。对 Kinesis Data Analytics 的所有 API 调用通过传输层安全性 (TLS) 进行保护，并通过 IAM 进行身份验证。客户端必须支持 TLS 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 Amazon Security Token Service (Amazon STS) 生成临时安全凭证来对请求进行签名。

Kinesis Data Analytics for Apache Flink

Amazon Kinesis Data Analytics 提供了在您开发和实施自己的安全策略时需要考虑的大量安全功能。以下最佳实践是一般指导原则，并不代表完整安全解决方案。由于这些最佳实践可能不适合您的环境或不满足您的环境要求，因此将其视为有用的考虑因素而不是惯例。

实施最低权限访问

在授予权限时，您可以决定谁获得哪些 Kinesis Data Analytics 资源的哪些权限。您可以对这些资源启用希望允许的特定操作。因此，您应仅授予执行任务所需的权限。实施最低权限访问对于减小安全风险以及可能由错误或恶意意图造成的影响至关重要。

使用 IAM 角色访问其他 Amazon 服务

您的 Kinesis Data Analytics 应用程序必须具有有效的凭证来访问其他服务中的资源，例如 Kinesis 数据流、Kinesis Data Firehose 传输流或 Amazon S3 存储桶。你不应该存储Amazon直接在该应用程序或

Amazon S3 存储桶中提供了凭证。这些是不会自动轮换的长期凭证，如果它们受到损害，可能会对业务产生重大影响。

相反，您应该使用 IAM 角色来管理应用程序的临时凭证以访问其他资源。在使用角色时，您不必使用长期凭证 (如用户名和密码或访问密钥) 来访问其他资源。

有关更多信息，请参阅 IAM 用户指南中的以下主题：

- IAM 角色
- 针对角色的常见情形：用户、应用程序和服务

实施从属资源中的服务器端加密

静态数据和传输中的数据在 Kinesis Data Analytics 中加密，并且无法禁用此加密。您应在您的从属资源中实施服务器端加密，例如 Kinesis 数据流、Kinesis Data Firehose 传输流和 Amazon S3 存储桶。有关在从属资源中实施服务器端加密的更多信息，请参阅 数据保护 (p. 197)。

使用 CloudTrail 监控 API 调用

Kinesis Data Analytics 与Amazon CloudTrail，这是在 Kinesis Data Analytics 中提供用户、角色或 Amazon 服务所采取操作的记录的服务。

使用收集的信息 CloudTrail，您可以确定向 Kinesis Data Analytics 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

有关更多信息，请参阅 the section called “使用 Amazon CloudTrail” (p. 234)。

Amazon Kinesis Data Analytics for Apache Flink

监控是保持 Amazon Kinesis Data Analytics 和您的 Kinesis 数据分析应用程序的可靠性、可用性和性能的重要组成部分。您应从 Amazon 解决方案的所有部分收集监控数据，以便更轻松地调试出现的多点故障。

在开始监控 Kinesis Data Analytics 之前，您应该制定一个监控计划，其中包括以下问题的答案：

- 监控目的是什么？
- 您将监控哪些资源？
- 监控这些资源的频率如何？
- 您将使用哪些监控工具？
- 谁负责执行监控任务？
- 出现错误时应通知谁？

下一步是在您的环境中为正常 Kinesis Data Analytics 性能设置基准。您可以通过在不同时间和不同负载条件下衡量性能来获得这一基准。当您监控 Kinesis Data Analytics 时，可以存储历史监控数据。然后，您可以将其与当前性能数据进行比较，确定正常的性能模式和性能异常，并找出解决问题的方法。

主题

日志记录

日志记录对于生产应用程序了解错误和失败非常重要。但是，日志子系统需要收集日志条目并将其转发到 CloudWatch 日志虽然有些日志记录很好而且很理想，但大量的日志记录可能会使服务超负荷并导致 Flink 应用程序落后。记录异常和警告当然是个好主意。但是您无法为 Flink 应用程序处理的每条消息生成日志消息。Flink 针对高吞吐量和低延迟进行了优化，但日志子系统不是。如果确实需要为每条已处理的消息生成日志输出，请使用额外的 DataStream 在 Flink 应用程序中，还有一个适当的接收器来将数据发送到 Amazon S3 或 CloudWatch. 请勿将 Java 日志记录系统用于此目的。此外，Kinesis Data Analytics`Debug` `Monitoring Log Level`设置会产生大量流量，这可能会产生反压。您只能在积极调查应用程序问题时使用它。

查询日志 CloudWatch 日志分析

CloudWatch Logs Insights 是一项功能强大的服务，可大规模查询日志。客户应利用其功能快速搜索日志，以识别和缓解运营事件期间的错误。

以下查询会在所有任务管理器日志中查找异常，并根据异常发生的时间对它们进行排序。

```
fields @timestamp, @message
| filter isPresent(throwableInformation.0) or isPresent(throwableInformation) or @message
 like /(Error|Exception)/
```

```
| sort @timestamp desc
```

有关其他有用查询，请参阅示例查询.

监控

在生产环境中运行流式处理应用程序时，您需要持续无限期地执行应用程序。对所有组件实施监控和适当报警是至关重要的，而不仅仅是 Flink 应用程序。否则，你可能会在早期错过新出现的问题，只有在运营事件完全解散且难以缓解时才意识到它。要监控的常规内容包括：

- 来源是否在摄取数据？
- 数据是否从源读取（从源的角度）？
- Flink 应用程序是否正在接收数据？
- Flink 应用程序能否跟上步伐，还是落后了？
- Flink 应用程序是否将数据保留到接收器中（从应用程序的角度来看）？
- 接收器是否正在接收数据？

然后，应该为 Flink 应用程序考虑更具体的指标。该CloudWatch 仪表板提供了一个好起点。有关要为生产应用程序监控哪些指标的更多信息，请参阅使用 CloudWatch 使用 Amazon Kinesis Data Analytics 发出警报 (p. 226). 这些指标包括：

- records_lag_max和millsBehindLatest— 如果应用程序使用的是 Kinesis 或 Kafka，则这些指标会指示应用程序是否落后，是否需要扩展以跟上当前的负载。这是一个很好的通用指标，可以轻松跟踪所有类型的应用程序。但是它只能用于响应式扩展，即当应用程序已经落后时。
- cpuUtilization和heapMemoryUtilization— 这些指标可以很好地表明应用程序的总体资源利用率，除非应用程序受到 I/O 限制，否则可用于主动扩展。
- 停机时间— 停机时间大于零表示应用程序出现故障。如果该值大于 0，则应用程序不处理任何数据。
- lastCheckpointSize和lastCheckpointDuration— 这些指标监控状态下存储了多少数据以及执行检查点需要多长时间。如果检查点增加或需要很长时间，应用程序将持续花费时间进行检查点操作，而实际处理的周期会减少。在某些时候，检查点可能会增加或需要很长时间以至于失败。除了监控绝对值外，客户还应考虑使用以下方法监控变化率`RATE(lastCheckpointSize)`和`RATE(lastCheckpointDuration)`.
- numberOfFailed检查点— 此指标计算自应用程序启动以来失败的检查点数。根据应用程序的不同，如果检查点偶尔失败，这是可以容忍的。但是，如果检查点经常出现故障，则应用程序可能不健康，需要进一步关注。我们建议监控`RATE(numberOfFailedCheckpoints)`警报梯度而不是绝对值。

设置应用程序日志记录

通过添加亚马逊 CloudWatch 日志记录选项到 Kinesis Data Analytics 应用程序，您可以监控应用程序事件或配置问题。

此主题介绍了如何配置应用程序，以将应用程序事件写入到 CloudWatch 日志流。一个 CloudWatch 日志记录选项是一个应用程序设置和权限集合，应用程序使用这些设置和权限配置它将应用程序事件写入到的方式 CloudWatch 日志。你可以添加和配置 CloudWatch 日志记录选项使用Amazon Web Services Management Console或者Amazon Command Line Interface(Amazon CLI)。

添加时，注意以下事项 CloudWatch 日志记录选项到您的应用程序：

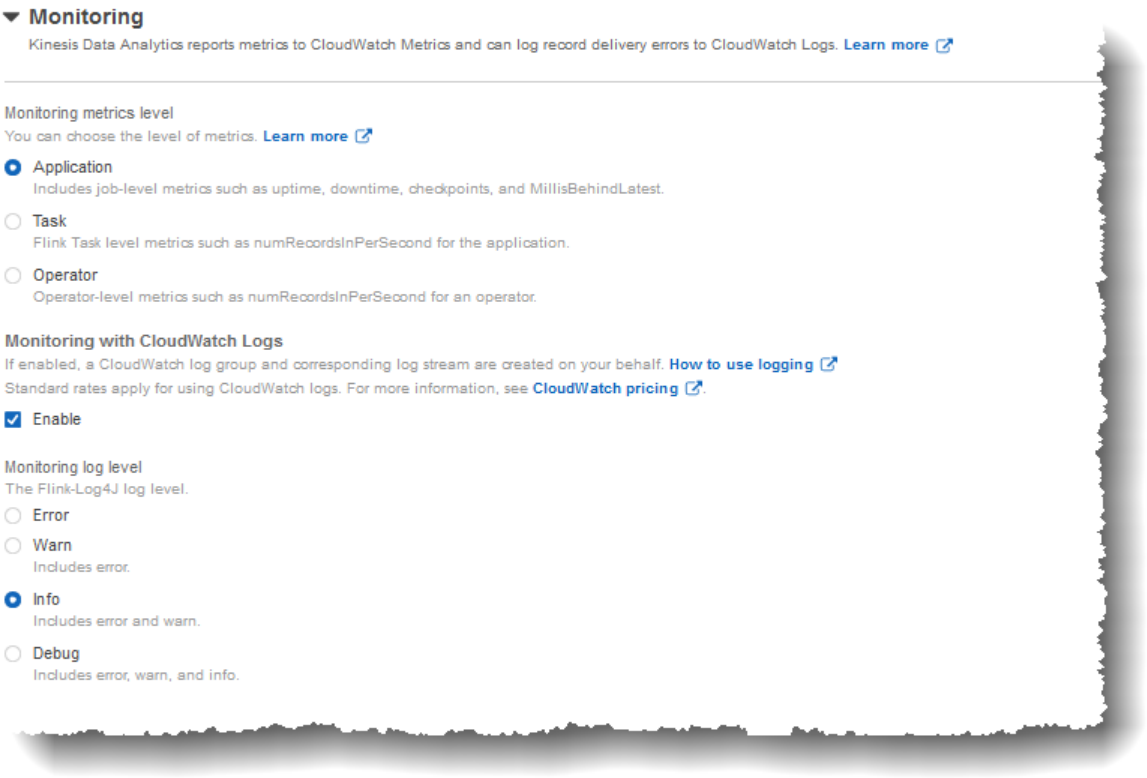
- 当你添加一个 CloudWatch 日志记录选项使用控制台，Kinesis Data Analytics 会创建 CloudWatch 日志组和日志流，并添加应用程序写入到日志流所需的权限。
- 当你添加一个 CloudWatch 日志记录选项时，您还必须创建应用程序的日志组和日志流，并添加应用程序写入到日志流所需的权限。

本主题包含下列部分：

设置 CloudWatch 使用控制台日志记录

当你启用 CloudWatch 在控制台中记录您的应用程序， CloudWatch 日志组和日志流已启用。此外，还会使用写入到流的权限更新应用程序的权限策略。

以下屏幕截图显示了CloudWatch 记录设置在配置应用程序页.



Kinesis Data Analytics 创建一个使用以下约定命名的日志组，其中*ApplicationName*是您的应用程序的名称。

```
/aws/kinesis-analytics/ApplicationName
```

Kinesis Data Analytics 使用以下名称在新日志组中创建一个日志流。

```
kinesis-analytics-log-stream
```

您可以使用设置应用程序监视指标级别和监控日志级别监控日志级别的 部分配置应用程序页. 有关应用程序日志级别的信息，请参阅the section called “应用程序监控级别” (p. 210)。

设置 CloudWatch 使用 CLI 日志记录

添加 CloudWatch 使用日志记录选项Amazon CLI中，执行以下操作：

- 创建 CloudWatch 日志组和日志流。
- 使用创建应用程序时添加日志记录选项CreateApplication操作或添加日志组成的现有应用程序添加日志记录选项AddApplicationCloudWatchLoggingOptionaction.
- 在应用程序的策略中添加权限以写入到日志。

本节包含以下主题：

创建 CloudWatch 日志组和日志流

你创建了一个 CloudWatch 日志组和流 CloudWatch 日志控制台或 API。有关创建的信息 CloudWatch 日志组和日志流，请参阅使用日志组和日志流.

使用应用程序 CloudWatch 日志记录选项

使用以下 API 操作来添加 CloudWatch 将日志选项记录到新的或现有的应用程序中，或者更改现有应用程序的日志选项。有关如何将 JSON 文件用于 API 操作输入的信息，请参阅 Kinesis Data Analytics API 示例代码 (p. 341)。

添加组 CloudWatch 在创建应用程序时记录选项

以下示例了如何使用`CreateApplication`操作来添加 CloudWatch 创建应用程序时的 log 选项。在此示例中*`# Amazon #### (ARN) CloudWatch ##############`*使用资源。有关该操作的更多信息，请参阅`CreateApplication`。

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "test-application-description",
    "RuntimeEnvironment": "FLINK-1_13",
    "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation":{
                            "BucketARN": "arn:aws:s3:::mybucket",
                            "FileKey": "myflink.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        }
    },
    "CloudWatchLoggingOptions": [{
      "LogStreamARN": "<Amazon Resource Name (ARN) of the CloudWatch log stream to add to
 the new application>"
 }]
}
```

添加组 CloudWatch 将选项记录到现有应用程序

以下示例了如何使用AddApplicationCloudWatchLoggingOption操作来添加 CloudWatch 日志选项到现有应用程序。在该示例中，将每个*#######*替换为您自己的信息。有关该操作的更多信息，请参阅AddApplicationCloudWatchLoggingOption。

```
{
   "ApplicationName": "<Name of the application to add the log option to>",
   "CloudWatchLoggingOption": {
      "LogStreamARN": "<ARN of the log stream to add to the application>"
   },
   "CurrentApplicationVersionId": <Version of the application to add the log to>
}
```

更新现有的 CloudWatch 日志选项

以下示例了如何使用UpdateApplication操作以修改现有的 CloudWatch 日志选项。在该示例中，将每个*#######*替换为您自己的信息。有关该操作的更多信息，请参阅 UpdateApplication。

```
{
   "ApplicationName": "<Name of the application to update the log option for>",
   "CloudWatchLoggingOptionUpdates": [
         {
            "CloudWatchLoggingOptionId": "<ID of the logging option to modify>",
            "LogStreamARNUpdate": "<ARN of the new log stream to use>"
         }
      ],
   "CurrentApplicationVersionId": <ID of the application version to modify>
}
```

删除集群 CloudWatch 应用程序中的日志选项

以下示例了如何使用DeleteApplicationCloudWatchLoggingOption删除现有的 CloudWatch 日志选项。在该示例中，将每个*#######*替换为您自己的信息。有关该操作的更多信息，请参阅DeleteApplicationCloudWatchLoggingOption。

```
{
   "ApplicationName": "<Name of application to delete log option from>",
   "CloudWatchLoggingOptionId": "<ID of the application log option to delete>",
   "CurrentApplicationVersionId": <Version of the application to delete the log option
 from>
}
```

设置应用程序日志记录级别

要设置应用程序日志记录的级别，请使用MonitoringConfigurationParameterCreateApplication动作或MonitoringConfigurationUpdateParameterUpdateApplicationaction.

有关应用程序日志级别的信息，请参阅the section called “应用程序监控级别” (p. 210)。

在创建应用程序时设置应用程序日志记录级别

CreateApplication 操作的以下示例请求将应用程序日志级别设置为 INFO。

```
{
   "ApplicationName": "MyApplication",
   "ApplicationDescription": "My Application Description",
   "ApplicationConfiguration": {
      "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType":"ZIPFILE"
      },
      "FlinkApplicationConfiguration":
         "MonitoringConfiguration": {
            "ConfigurationType": "CUSTOM",
            "LogLevel": "INFO"
         }
      },
   "RuntimeEnvironment": "FLINK-1_13",
   "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

更新应用程序日志记录级别

`UpdateApplication` 操作的以下示例请求将应用程序日志级别设置为 `INFO`。

```
{
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "MonitoringConfigurationUpdate": {
            "ConfigurationTypeUpdate": "CUSTOM",
            "LogLevelUpdate": "INFO"
         }
      }
   }
}
```

将写入权限添加到 CloudWatch 日志流

Kinesis Data Analytics 需要权限才能将配置错误写入到中 CloudWatch. 您可以将这些权限添加到Amazon Identity and Access Management(IAM) 角色由 Kinesis Data Analytics 担任。

有关将 IAM 角色用于 Kinesis Data Analytics 的更多信息，请参阅Amazon Kinesis Data Analytics for Apache Flink (p. 198).

信任策略

要向 Kinesis Data Analytics 授予权限以担任 IAM 角色，您可以将以下信任策略附加到服务执行角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

权限策略

为应用程序授予权限以将日志事件写入到 CloudWatch 在 Kinesis Data Analytics 资源中，您可以使用以下 IAM 权限策略。为日志组和流提供正确的 Amazon 资源名称 (ARN)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt0123456789000",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:my-log-group:log-stream:my-
log-stream*",
                "arn:aws:logs:us-east-1:123456789012:log-group:my-log-group:*",
                "arn:aws:logs:us-east-1:123456789012:log-group:*",
            ]
        }
    ]
}
```

应用程序监控级别

您可以使用应用程序的监控指标级别和监控日志级别，以控制生成应用程序日志消息的过程。

应用程序的监控指标级别控制日志消息的粒度。监控指标级别定义如下：

- 应用程序：指标范围是整个应用程序。
- 任务：指标范围是每个任务。有关任务的信息，请参阅the section called “扩缩” (p. 27)。
- 运算符：指标范围是每个操作符。有关操作符的信息，请参阅the section called “DataStreamAPI 操作符” (p. 12)。
- Parallelism：指标范围是应用程序并行度。您只能使用 MonitoringConfigurationUpdateParameter UpdateApplicationAPI。您无法使用控制台设置该指标级别。有关并行度的信息，请参阅the section called “扩缩” (p. 27)。

应用程序的监控日志级别控制应用程序日志的详细程度。监控日志级别定义如下：

- 错误：应用程序的潜在灾难性事件。
- 警告：应用程序的可能有害情况。
- Info：应用程序的信息性和暂时性故障事件。我们建议您使用该日志记录级别。
- Debug：对调试应用程序非常有用的精细信息性事件。注意：仅将该级别用于临时调试目的。

日志记录最佳实践

我们建议您的应用程序使用信息日志记录级别。我们建议您使用该级别，以确保您看到 Apache Flink 错误，这些错误是在信息级别而不是错误级别记录的。

我们建议您仅在调查应用程序问题时临时使用调试级别。在解决问题后，请切换回信息级别。使用调试日志记录级别将严重影响应用程序的性能。

过多的日志记录也可能会严重影响应用程序性能。例如，我们建议您不要为每个处理的记录写入一个日志条目。过多的日志记录可能会导致严重的数据处理瓶颈，并且可能会导致从源中读取数据时出现反向压力。

日志记录故障排除

如果没有将应用程序日志写入到日志流，请验证以下内容：

- 验证应用程序的 IAM 角色和策略是否正确。应用程序的策略需要具有以下权限以访问日志流：
  - `logs:PutLogEvents`
  - `logs:DescribeLogGroups`
  - `logs:DescribeLogStreams`

  有关更多信息，请参阅 the section called “将写入权限添加到 CloudWatch 日志流” (p. 209)。
- 验证应用程序是否正在运行。要检查应用程序的状态，请在控制台中查看应用程序的页面，或者使用DescribeApplication要么ListApplications行动。
- 显示器 CloudWatch 指标，如`downtime`以诊断其他应用程序问题。有关阅读的信息 CloudWatch 指标，请参阅指标与维度 (p. 214).

下一个步骤

启用后 CloudWatch 登录你的应用程序，你可以使用 CloudWatch 记录 Insights 以分析您的应用程序日志。有关更多信息，请参阅 the section called “分析日志” (p. 211)。

使用分析日志 CloudWatch 日志分析

在你添加了 CloudWatch 日志记录选项，如上一部分中所述，您可以使用 CloudWatch 记录 Insights 以查询日志流中的特定事件或错误。

CloudWatch 通过使用 Logs Insights，您可以交互地搜索和分析中的日志数据 CloudWatch 日志。

有关入门的信息 CloudWatch 日志分析，请参阅使用 分析日志数据 CloudWatch 日志分析.

运行示例查询

本部分描述如何运行示例 CloudWatch 日志分析查询。

先决条件

- 在中设置了现有的日志组和日志流 CloudWatch 日志。
- 在中存储了现有的日志 CloudWatch 日志。

如果您使用诸如Amazon CloudTrail、Amazon Route 53 或 Amazon VPC，您可能已从这些服务中设置日志以发送到 CloudWatch 日志。有关将日志发送到的更多信息，请参阅 CloudWatch 日志，请参阅入门 CloudWatch 日志.

中的查询 CloudWatch Logs Insightics 返回日志事件中的一组字段，或返回对日志事件执行的数学聚合或其他操作的结果。本节说明了一个返回一组日志事件的查询。

运行 CloudWatch Insight示例查询

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

3. 屏幕顶部附近的查询编辑器包含一个默认查询，它返回 20 个最近的日志事件。在查询编辑器上方，选择一个要查询的日志组。

   选择日志组时， CloudWatch Logs Insights 会自动检测日志组数据中的字段，并将其显示在发现的字段在右窗格中。它还显示此日志组中的日志事件随时间变化的条形图。该条形图显示与您的查询和时间范围匹配的日志组中的事件分布情况，而不仅仅是表中显示的事件。
4. 选择 Run query (运行查询)。

   显示此查询的结果。在本示例中，结果是任何类型的最新 20 个日志事件。
5. 要查看某个返回的日志事件的所有字段，请选择该日志事件左侧的箭头。

有关如何运行和修改的更多信息 CloudWatch 记录见解查询，请参阅运行和修改示例查询.

示例查询

本节包括 CloudWatch 记录用于分析 Kinesis Data Analytics 应用程序日志的示例查询。这些查询搜索一些示例错误情况，并作为模板以编写查找其他错误情况的查询。

> **Note**
>
> 替换区域 (*us-west-2*)、账户 ID (*012345678901*) 和应用程序名称 (*YourApplication*) 和您的账户 ID，包括应用程序的区域和您的账户 ID。

本主题包含下列部分：

- 分析操作：分配任务 (p. 212)
- 分析操作：并行变化 (p. 213)
- 分析错误：拒绝访问 (p. 213)
- 分析错误：找不到源或接收器 (p. 213)
- 分析错误：应用程序的任务相关故障 (p. 213)

分析操作：分配任务

以下 CloudWatch Logs Insightics 查询返回 Apache Flink Job 管理器在任务管理器之间分配的任务数。您需要设置查询的时间范围以与某个作业运行匹配，以便查询不会返回以前作业的任务。有关并行度的更多信息，请参阅扩缩 (p. 27)。

```
fields @timestamp, message
| filter message like /Deploying/
| parse message " to flink-taskmanager-*" as @tmid
| stats count(*) by @tmid
| sort @timestamp desc
| limit 2000
```

以下 CloudWatch 日志见解查询返回分配给每个任务管理器的子任务。子任务总数是每个任务的并行度的总和。任务并行度来自于操作符并行度，默认情况下，它与应用程序的并行度相同，除非您在代码中指定 `setParallelism` 以对其进行更改。有关设置运算符并行的更多信息，请参阅设置并行度：操作员级别中的Flink 文档.

```
fields @timestamp, @tmid, @subtask
| filter message like /Deploying/
| parse message "Deploying * to flink-taskmanager-*" as @subtask, @tmid
| sort @timestamp desc
```

```
| limit 2000
```

有关任务调度的更多信息，请参阅作业和计划中的Flink 文档.

分析操作：并行变化

以下 CloudWatch Logs Insights 查询返回对应用程序并行度的更改（例如，由于自动扩展）。该查询还会返回对应用程序并行度的手动更改。有关自动扩展的更多信息，请参阅the section called “自动扩展” (p. 29)。

```
fields @timestamp, @parallelism
| filter message like /property: parallelism.default, /
| parse message "default, *" as @parallelism
| sort @timestamp asc
```

分析错误：拒绝访问

以下 CloudWatch Insight查询返回`Access Denied`日志。

```
fields @timestamp, @message, @messageType
| filter applicationARN like /arn:aws:kinesisanalytics:us-west-2:012345678901:application
\/YourApplication/
| filter @message like /AccessDenied/
| sort @timestamp desc
```

分析错误：找不到源或接收器

以下 CloudWatch Insight查询返回`ResourceNotFound`日志。`ResourceNotFound`如果找不到 Kinesis 源或接收器，将记录结果。

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalytics:us-west-2:012345678901:application
\/YourApplication/
| filter @message like /ResourceNotFoundException/
| sort @timestamp desc
```

分析错误：应用程序的任务相关故障

以下 CloudWatch 日志Insights 查询返回应用程序的任务相关故障日志志志志志志。如果应用程序状态从`RUNNING`到`RESTARTING`.

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalytics:us-west-2:012345678901:application
\/YourApplication/
| filter @message like /switched from RUNNING to RESTARTING/
| sort @timestamp desc
```

对于使用 Apache Flink 版本 1.8.2 及更低版本的应用程序，与任务相关的故障将导致应用程序状态从`RUNNING`到`FAILED`相反。使用 Apache Flink 1.8.2 及更早版本时，请使用以下查询来搜索与应用程序任务相关的故障：

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalytics:us-west-2:012345678901:application
\/YourApplication/
| filter @message like /switched from RUNNING to FAILED/
| sort @timestamp desc
```

查看 Kinesis Data Analytics 指标和维度

本主题包含下列部分：

在适用于 Apache Flink 的 Kinesis Data Analytics 应用程序处理数据源时，Kinesis Data Analts Antalyts 向 Apache Flink 应用程序处理数据源时 CloudWatch.

应用程序指标

Important

`containerCPUUtilization`、`containerMemoryUtilization`和`containerDiskUtilization`是新的指标，将在 CloudWatch 应用程序重新启动或更新之后。

指标	Unit	描述	Level	使用说明	
`backPressuredTimeMsPerSecond`	毫秒	此任务或操作员每秒回压的时间（以毫秒为单位）。	任务、操作符、并行度	*适用于运行 Flink 1.13 的 KDA 应用程序。 这些指标可用于识别应用程序中的瓶颈。	
`busyTimeMsPerSecond`	毫秒	此任务或操作员每秒忙碌的时间（以毫秒为单位）（既没有空闲也没有后压）。如果无法计算该值，则可以为 NaN。	任务、操作符、并行度	*适用于运行 Flink 1.13 的 KDA 应用程序。 这些指标可用于识别应用程序中的瓶颈。	
`cpuUtilization`	百分比	任务管理器的 CPU 使用百分率。例如，如果有五个任务管理器，Kinesis Data Analytics 会在每个报告间隔发布此指标的五个样本。	应用程序	您可以使用此指标来监视应用程序中的最低、平均和最大 CPU 使用率。	
`containerCPUUtilization`	百分比	Flink 应用程序集群中任务管理器容器的 CPU 使用	应用程序	每个容器的计算公式为：	

指标	Unit	描述	Level	使用说明	
		率总体百分比。例如，如果有五个任务管理器，则相应地有五个 TaskManager 容器和 Kinesis Data Analytics 每 1 分钟报告间隔发布此指标的 2 * 5 个样本。		容器消耗的总 CPU 时间（以秒为单位）* 100 / 容器 CPU 限制（以 CPU/秒为单位） 这些区域有：`CPUUtilization`指标仅考虑了 CPU 使用率 TaskManager JVM 进程在容器内运行。在同一个容器中，还有其他组件在 JVM 之外运行。这些区域有：`containerCPUUtilization`metric 可以让您更全面地了解容器的 CPU 耗尽情况以及由此导致的故障，包括所有进程。	

指标	Unit	描述	Level	使用说明	
containerMemoryUtilization	百分比	Flink 应用程序集群中任务管理器容器的总体内存使用百分比。例如，如果有五个任务管理器，则相应地有五个 TaskManager 容器和 Kinesis Data Analytics 每 1 分钟报告间隔发布此指标的 2 * 5 个样本。	应用程序	每个容器的计算公式为： 容器内存使用量（字节）* 100 / 根据 pod 部署规范的容器内存限制（以字节为单位） 这些区域有：HeapMemoryUtilization和Manag 标仅考虑特定的内存指标，如堆内存使用量 TaskManager JVM 或托管内存（本机进程在 JVM 之外的内存使用情况，如Rocksdb 状态后端)。这些区域有：containerMemoryUtilizationM 通过包含工作集内存为您提供更全面的画面，这是内存总耗尽的更好跟踪器。精疲力尽后，它会导致Out of Memory Error(对于) TaskManager Pod。	
containerDiskUtilization	百分比	Flink 应用程序集群中任务管理器容器中磁盘利用率的总体百分比。例如，如果有五个任务管理器，则相应地有五个 TaskManager 容器和 Kinesis Data Analytics 每 1 分钟报告间隔发布此指标的 2 * 5 个样本。	应用程序	每个容器的计算公式为： 磁盘使用量（以字节为单位）* 100 /容器的磁盘限制（以字节为单位） 对于容器，它表示容器根卷所在的文件系统的利用率。	
currentInputWatermark	毫秒	此应用程序/操作员/任务/线程收到的最后一个水印	应用程序、操作员、任务、并行度	仅为具有两个输入的维度发出该记录。这是上次收到的水印的最小值。	

指标	Unit	描述	Level	使用说明	
currentOutputWatermark	毫秒	此应用程序/操作员/任务/线程发出的最后一个水印	应用程序、操作员、任务、并行度		
downtime	毫秒	对于当前处于故障/恢复状态的作业，在该中断期间经过的时间。	应用程序	该指标测量作业发生故障或恢复时经过的时间。该指标为运行的作业返回 0，并为完成的作业返回 -1。如果该指标不是 0 或 -1，则表明应用程序的 Apache Flink 作业无法运行。	
fullRestarts	计数	此作业自提交以来完全重新启动的总次数。此指标不衡量精细的重启。	应用程序	您可以使用此指标来评估一般应用程序的运行状况。在 Kinesis Data Analytics 的内部维护期间，可能会重新启动。重新启动的频率可能表示应用程序出现问题。	
heapMemoryUtilization	百分比	任务管理器中的整体堆内存利用率。例如，如果有五个任务管理器，Kinesis Data Analytics 会在每个报告间隔发布此指标的五个样本。	应用程序	您可以使用此指标来监视应用程序中的最小、平均和最大堆内存利用率。使用以下公式为所有任务管理器计算该值： `(Heap.Used / Heap.Committed)`	
idleTimeMsPerSecond*	毫秒	此任务或运算符每秒空闲（没有要处理的数据）的时间（以毫秒为单位）。空闲时间不包括向后加压时间，因此，如果任务受到回压，则不会处于空闲状态。	任务、操作符、并行度	*适用于运行 Flink 1.13 的 KDA 应用程序。 这些指标可用于识别应用程序中的瓶颈。	

指标	Unit	描述	Level	使用说明	
lastCheckpointSize	字节	上一个检查点的总大小	应用程序	您可以使用该指标确定运行的应用程序存储使用率。 如果该指标的值不断增加，则可能表明应用程序出现问题，例如内存泄漏或瓶颈。	
lastCheckpointDuration	毫秒	完成上一个检查点所花的时间	应用程序	该指标测量完成最近的检查点所花的时间。如果该指标的值不断增加，则可能表明应用程序出现问题，例如内存泄漏或瓶颈。在某些情况下，您可以禁用检查点以解决该问题。	
managedMemoryUsed*	字节	当前使用的托管内存量。	应用程序、操作员、任务、并行度	*适用于运行 Flink 1.13 的 KDA 应用程序。 这与由 Flink 在 Java 堆之外管理的内存有关。它用于 RocksDB 状态后端，也可用于应用程序。	
managedMemoryTotal*	字节	托管内存的总量。	应用程序、操作员、任务、并行度	*适用于运行 Flink 1.13 的 KDA 应用程序。 这与由 Flink 在 Java 堆之外管理的内存有关。它用于 RocksDB 状态后端，也可用于应用程序。	
managedMemoryUtilization*	百分比	派生 managedMemoryUsed/managedMemoryTotal	应用程序、操作员、任务、并行度	*适用于运行 Flink 1.13 的 KDA 应用程序。 这与由 Flink 在 Java 堆之外管理的内存有关。它用于 RocksDB 状态后端，也可用于应用程序。	

指标	Unit	描述	Level	使用说明	
numberOfFailedCheckpoints	计数	检查点操作失败的次数。	应用程序	您可以使用此指标来监视应用程序的运行状况和进度。检查点可能由于应用程序问题（例如吞吐量或权限问题）而失败。	
numRecordsIn	计数	该应用程序、操作符或任务收到的总记录数。	应用程序、操作员、任务、并行度	度量的级别指定此指标是衡量整个应用程序、特定操作员还是特定任务已收到的记录总数。	
numRecordsInPerSecond	计数/秒	该应用程序、操作符或任务每秒收到的总记录数。	应用程序、操作员、任务、并行度	度量的 “级别” 指定此指标是否衡量整个应用程序、特定操作员或特定任务每秒收到的记录总数。	
numRecordsOut	计数	该应用程序、操作符或任务发出的总记录数。	应用程序、操作员、任务、并行度	该指标的 “级别” 指定此指标是否衡量整个应用程序、特定操作员或特定任务发出的记录总数。	
numLateRecordsDropped	计数	由于到达延迟，该操作符或任务丢弃的记录数。	应用程序、操作员、任务、并行度		
numRecordsOutPerSecond	计数/秒	该应用程序、操作符或任务每秒发出的总记录数。	应用程序、操作员、任务、并行度	该指标的 “级别” 指定此指标是否衡量整个应用程序、特定操作员或特定任务每秒发出的记录总数。	
oldGenerationGCCount	计数	所有任务管理器中发生的旧垃圾回收操作的总数。	应用程序		
oldGenerationGCTime	毫秒	执行旧的垃圾回收操作所花费的总时间。	应用程序	您可以使用此度量来监视垃圾回收时间的总和、平均和最大值。	

指标	Unit	描述	Level	使用说明	
threadCount	计数	应用程序使用的活动线程总数。	应用程序	该指标用于测量应用程序代码使用的线程的数目。这与应用程序并行度不同。	
uptime	毫秒	作业无中断运行的时间。	应用程序	您可以使用该指标确定作业是否成功运行。此指标为完成的作业返回 -1。	

Kinesis Data Streams

Amazon除了以下内容以外，还发出 Kinesis Data Streams 的所有记录：

指标	Unit	描述	Level	使用说明
millisBehindLatest	毫秒	使用者落后流开头的毫秒数，表示使用者落后当前时间有多远。	应用程序（用于流），并行度（用于 ShardId）	• 值为 0 表示记录处理是同步的，并且此时没有要处理的新记录。可以按流名称和分片 ID 来指定特定分片的指标。 • 值为 -1 表示该服务尚未报告指标值。
bytesRequestedPerFetch	字节	在一次 getRecords 调用中请求的字节数。	应用程序（用于流），并行度（用于 ShardId）	

Amazon MSK 连接器指标

Amazon除了以下内容以外，还发出亚马逊 MSK 的所有记录：

指标	Unit	描述	Level	使用说明
currentoffsets	不适用	使用者的当前读取偏移量（对于每个分区）。可以按主题名称和分区 ID 来指定特定分区的指标。	应用程序（针对主题）、并行度（用于 PartitionId）	
commitsFailed	不适用	Kafka 偏移提交失败的总数（如果启用了偏移提交和检查点功能）。	应用程序、操作员、任务、并行度	向 Kafka 提交偏移量只是暴露消费者进度的一种手段，因此提交失败不会

指标	Unit	描述	Level	使用说明
				影响 Flink 的检查点分区偏移的完整性。
commitsSucceeded	不适用	如果启用了偏移提交和检查点操作，则成功提交到 Kafka 的总偏移量。	应用程序、操作员、任务、并行度	
committedoffsets	不适用	上次成功提交的 Kafka 偏移量（对于每个分区）。可以按主题名称和分区 ID 来指定特定分区的指标。	应用程序（针对主题）、并行度（用于 PartitionId）	
records_lag_max	计数	最大延迟，以该窗口中的任何分区的记录数表示	应用程序、操作员、任务、并行度	
bytes_consumed_rate	字节	主题平均每秒使用的字节数	应用程序、操作员、任务、并行度	

Apache Zeppelelelin 指标

对于 Studio 笔记本Amazon在应用程序级别发出以下指标：KPUs、cpuUtilization、heapMemoryUtilization、oldGenerationGCTime、oldGenerationGCCount, 和threadCount. 此外，它还会在应用程序级别发出下表中显示的指标。

指标	Unit	描述	Prometheus 的名字
zeppelinCpuUtilization	百分比	Apache Zeppelin 服务器中 CPU 利用率的总体百分比。	process_cpu_usage
zeppelinHeapMemoryUtilization	百分比	Apache Zeppelin 服务器的堆内存利用率的总体百分比。	jvm_memory_used_bytes
zeppelinThreadCount	计数	Apache Zeppelin 服务器使用的实时线程总数。	jvm_threads_live_threads
zeppelinWaitingJobs	计数	排队等待线程的 Apache Zeppelin 作业的数量。	jetty_threads_jobs
zeppelinServerUptime	秒	服务器启动和运行的总时间。	process_uptime_seconds

查看 CloudWatch 指标

您可以查看 CloudWatch 使用Amazon 应用程序的指标 CloudWatch 控制台或Amazon CLI.

使用 查看指标 CloudWatch 控制台

1. 打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/.
2. 在导航窗格中，选择 Metrics（指标）。
3. 在CloudWatch 按类别划分指标” 窗格中 Amazon Kinesis Data Analytics 请选择指标类别。
4. 在上方窗格中，滚动以查看完整指标列表。

使用 Amazon CLI 查看指标

- 在命令提示符处，使用以下命令。

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

设置 CloudWatch 指标报告级别

您可以控制应用程序创建的应用程序指标的级别。Kinesis Data Analytics for Apache Flink 支持以下指标级别：

- 应用程序：应用程序仅报告每个应用程序的最高指标级别。默认情况下，Kinesis Data Analytics 指标在应用程序级别发布。
- 任务：应用程序报告任务特定的指标的指标维度（例如，每秒在应用程序中传入和传出的记录数）。
- 运算符: 应用程序报告操作符指标报告级别定义的指标的操作符特定指标维度，例如，每个筛选或映射操作的指标。
- 并行度：该应用程序报告`Task`和`Operator`每个执行线程的级别的指标。由于成本过高，建议不要将该报告级别用于并行度设置高于 64 的应用程序。

  Note

  由于服务生成的度量数据量很大，您只能使用此度量级别进行故障排除。您只能使用 CLI 设置此指标级别。在控制台中不提供此指标等级。

默认级别是应用程序。应用程序报告当前级别和所有更高级别的指标。例如，如果报告级别设置为操作符，则应用程序报告应用程序、任务和操作符指标。

你设置了 CloudWatch 指标报告级别使用`MonitoringConfiguration`Parameter`CreateApplication`动作，或者`MonitoringConfigurationUpdate`Parameter`UpdateApplication`action. 以下示例请求`UpdateApplication`动作设置 CloudWatch 指标报告级别为任务：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "MonitoringConfigurationUpdate": {
            "ConfigurationTypeUpdate": "CUSTOM",
            "MetricsLevelUpdate": "TASK"
         }
      }
   }
}
```

您还可以使用配置日志记录级别`LogLevel`Parameter`CreateApplication`动作或`LogLevelUpdate`Parameter`UpdateApplication`action. 您可以使用以下日志级别：

- `ERROR`：记录潜在可恢复的错误事件。
- `WARN`：记录可能导致错误的警告事件。
- `INFO`：记录信息性事件。
- `DEBUG`：记录常规调试事件。

有关 Log4j 日志记录级别的更多信息，请参阅。自定义日志级别中的Apache Log4j文档中）。

将自定义指标与Amazon Kinesis Data Analytics 结合使用自定义指标

Kinesis Data Analytics CloudWatch，包括资源使用率和吞吐量的指标。此外，您可以创建自己的指标来跟踪特定于应用程序的数据，例如处理事件或访问外部资源。

本主题包含下列部分：

工作方式

Kinesis Data Analytics 中的自定义指标使用 Apache Flink 指标系统。Apache Flink 指标具有以下属性：

- 类型：指标的类型描述了它如何衡量和报告数据。可用的 Apache Flink 量度类型包括计数、量规、直方图和仪表。有关 Apache Flink 指标类型的更多信息，请参阅。指标类型.

  Note

  Amazon CloudWatch 指标不支持直方图 Apache Flink 量度类型。 CloudWatch 只能显示 Apache Flink 计数、仪表盘和计量器类型的指标。
- Scope: 指标的范围由其标识符和一组键值对组成，这些键值对指示指标将如何报告给 CloudWatch. 指标的标识符包含以下各项：
  - 系统范围，表示报告指标的级别（例如 Operator）。
  - 用户范围，用于定义诸如用户变量或指标组名称之类的属性。这些属性是使用定义的`MetricGroup.addGroup(key, value)`要么`MetricGroup.addGroup(name)`.

  有关指标范围的更多信息，请参阅范围.

有关 Apache Flink 指标的更多信息，请参阅。指标中的Flink 文档.

要在适用于 Apache Flink 的 Kinesis Data Analytics 应用程序中创建自定义指标，您可以从任何扩展的用户函数访问 Apache Flink 指标系统`RichFunction`通过调用`GetMetricGroup`. 该方法返回一个MetricGroup对象，您可以使用它来创建和注册自定义指标。Kinesis Data Analytics 报告使用组密钥创建的所有指标`KinesisAnalytics`到 CloudWatch. 您定义的自定义指标具有以下特征：

- 您的自定义指标有指标名称和组名称。这些名称必须由字母数字字符组成。
- 您在用户作用域中定义的属性（除了`KinesisAnalytics`量度组）发布为 CloudWatch 维度。
- 自定义指标发布在`Application`默认是级别。
- 维度（任务/运算符/并行度）根据应用程序的监视级别添加到指标中。您可以使用设置应用程序的监视级别MonitoringConfigurationParameterCreateApplication动作，或者或MonitoringConfigurationUpdateParameterUpdateApplicationaction.

示例

以下代码示例演示了如何创建映射类、如何创建和递增自定义指标，以及如何通过将映射类添加到DataStream对象。

记录计数自定义指标

以下代码示例演示如何创建一个映射类，该类用于创建对数据流中的记录进行计数的量度（功能与numRecordsIn指标):

```
    private static class NoOpMapperFunction extends RichMapFunction<String, String> {
        private transient int valueToExpose = 0;
        private final String customMetricName;

        public NoOpMapperFunction(final String customMetricName) {
            this.customMetricName = customMetricName;
        }

        @Override
        public void open(Configuration config) {
            getRuntimeContext().getMetricGroup()
                    .addGroup("kinesisanalytics")
                    .addGroup("Program", "RecordCountApplication")
                    .addGroup("NoOpMapperFunction")
                    .gauge(customMetricName, (Gauge<Integer>) () -> valueToExpose);
        }

        @Override
        public String map(String value) throws Exception {
            valueToExpose++;
            return value;
        }
    }
```

在上一示例中，valueToExpose变量为应用程序处理的每条记录递增。

定义映射类后，您可以创建一个实现映射的应用程序内流：

```
DataStream<String> noopMapperFunctionAfterFilter =
    kinesisProcessed.map(new NoOpMapperFunction("FilteredRecords"));
```

有关此应用程序的完整代码，请参阅记录计数自定义指标应用程序.

字数自定义指标

以下代码示例说明了如何创建映射类，以创建对数据流中的字数进行计数的量度：

```
private static final class Tokenizer extends RichFlatMapFunction<String, Tuple2<String,
 Integer>> {

            private transient Counter counter;

            @Override
            public void open(Configuration config) {
                this.counter = getRuntimeContext().getMetricGroup()
                        .addGroup("kinesisanalytics")
                        .addGroup("Service", "WordCountApplication")
                        .addGroup("Tokenizer")
                        .counter("TotalWords");
```

```
            }

            @Override
            public void flatMap(String value, Collector<Tuple2<String, Integer>>out) {
                // normalize and split the line
                String[] tokens = value.toLowerCase().split("\\W+");

                // emit the pairs
                for (String token : tokens) {
                    if (token.length() > 0) {
                        counter.inc();
                        out.collect(new Tuple2<>(token, 1));
                    }
                }
            }
        }
```

在上一示例中，counter对于应用程序处理的每个单词，变量都会递增。

定义映射类后，您可以创建一个实现映射的应用程序内流：

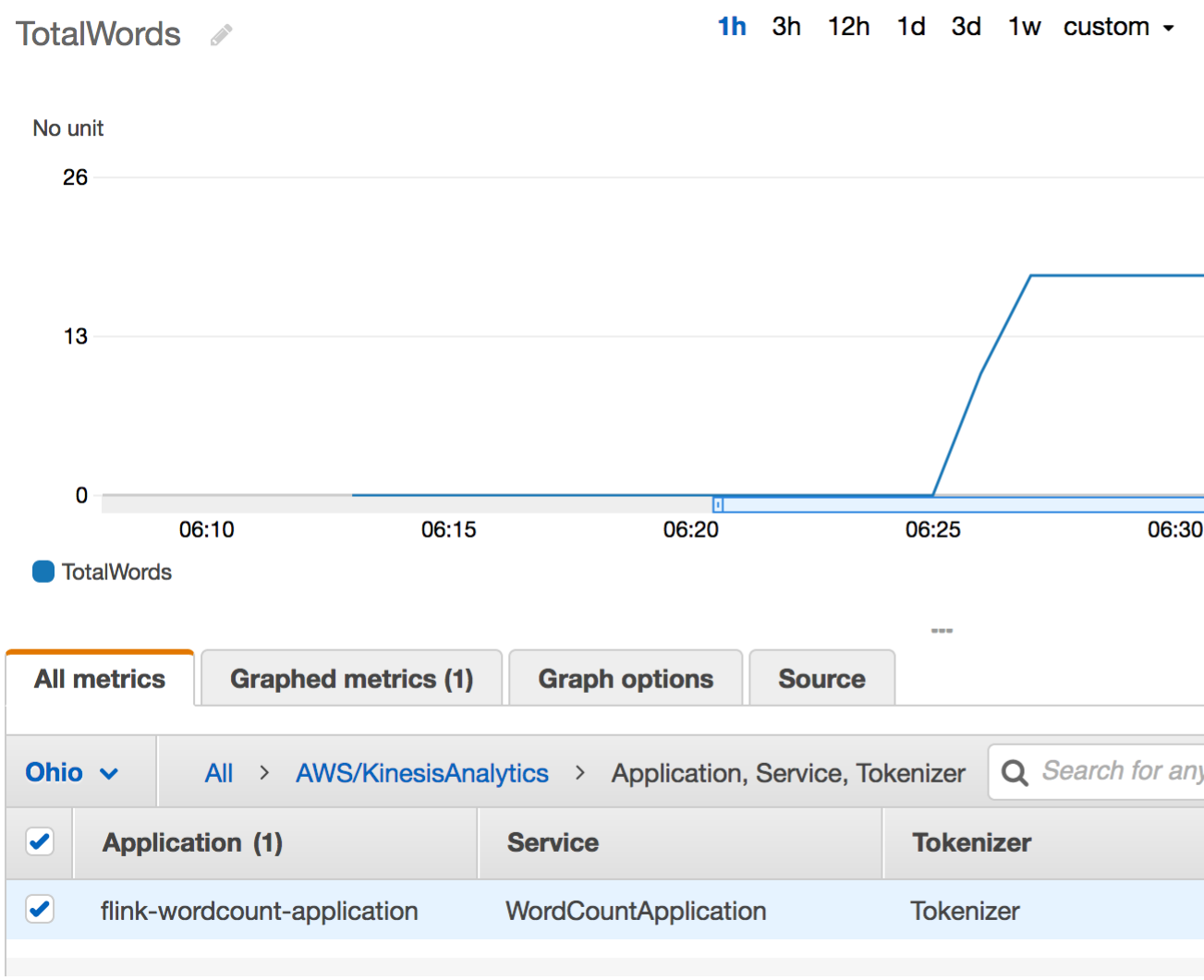
```
// Split up the lines in pairs (2-tuples) containing: (word,1), and
// group by the tuple field "0" and sum up tuple field "1"
DataStream<Tuple2<String, Integer>> wordCountStream = input.flatMap(new
 Tokenizer()).keyBy(0).sum(1);

// Serialize the tuple to string format, and publish the output to kinesis sink
wordCountStream.map(tuple -> tuple.toString()).addSink(createSinkFromStaticConfig());
```

有关此应用程序的完整代码，请参阅字数统计自定义指标应用程序.

查看自定义指标

应用程序的自定义指标将显示在 CloudWatch 中的指标控制台AWS/KinesisAnalytics控制面板，在应用程序指标组。



使用 CloudWatch 使用 Amazon Kinesis Data Analytics 发出警报

使用 Amazon CloudWatch 指标警报，你看 CloudWatch 您指定的时间段内的指标。告警根据指标或表达式在多个时间段内相对于某阈值的值执行一项或多项操作。操作的一个示例是将通知发送到 Amazon Simple Notification Service (Amazon SNS) 主题。

有关 的更多信息 CloudWatch 警报，请参阅使用 Amazon CloudWatch Alarms.

建议的 警报

本部分包含用于监控 Kinesis Data Analytics 应用程序的推荐警报。

此表包含以下列：

- 指标表达式：要根据阈值进行测试的度量或量度表达式。
- 统计数据：用于检查指标的统计数据，例如Average.

- 阈值：使用此警报要求您确定一个阈值，该阈值定义了应用程序预期性能的极限。您需要通过在正常条件下监控应用程序来确定此阈值。
- 描述：可能触发此警报的原因以及该情况的可能解决方案。

指标表达式	统计数据	Threshold	描述
#### > 0	Average	0	A downtime greater than zero indicates that the application has failed. If the value is larger than 0, the application is not processing any data. Recommended for all applications. The ### # metric measures the duration of an outage. A downtime greater than zero indicates that the application has failed. For troubleshooting, see 应用程序正在重启 (p. 320).
`# (numberOfFailed#` ### > 0	Average	0	This metric counts the number of failed checkpoints since the application started. Depending on the application, it can be tolerable if checkpoints fail occasionally. But if checkpoints are regularly failing, the application is likely unhealthy and needs further attention. We recommend monitoring RATE(numberOfFailedCheckpoints) to alarm on the gradient and not on absolute values. Recommended for all applications. Use this metric to monitor application health and checkpointing progress. The application saves state data to checkpoints when it's healthy. Checkpointing can fail due to timeouts if the application isn't making progress in processing the input data. For troubleshooting, see

指标表达式	统计数据	Threshold	描述
			Checkpointing 已超时 (p. 333).
## ##numRecordsOutPerSecond < threshold	Average	The minimum number of records emitted from the application during normal conditions.	Recommended for all applications. Falling below this threshold can indicate that the application isn't making expected progress on the input data. For troubleshooting, see 吞吐量太慢 (p. 321).
records_lag_max| millsBehindLatest > threshold	Maximum	The maximum expected latency during normal conditions.	If the application is consuming from Kinesis or Kafka, these metrics indicate if the application is falling behind and needs to be scaled in order to keep up with the current load. This is a good generic metric that is easy to track for all kinds of applications. But it can only be used for reactive scaling, i.e., when the application has already fallen behind. Recommended for all applications. Use the records_lag_max metric for a Kafka source, or the millisBehindLatest for a Kinesis stream source. Rising above this threshold can indicate that the application isn't making expected progress on the input data. For troubleshooting, see 吞吐量太慢 (p. 321).

指标表达式	统计数据	Threshold	描述
`lastCheckpointDuration` > threshold	Maximum	The maximum expected checkpoint duration during normal conditions.	Monitors how much data is stored in state and how long it takes to take a checkpoint. If checkpoints grow or take long, the application is continuously spending time on checkpointing and has less cycles for actual processing. At some points, checkpoints may grow so or take so long that they fail. In addition to monitoring absolute values, customers should also considering monitoring the change rate with `#(lastCheckpointSize#` and `#(lastCheckpointDuration#`. If the `lastCheckpointDuration` continuously increases, rising above this threshold can indicate that the application isn't making expected progress on the input data, or that there are problems with application health such as backpressure. For troubleshooting, see 无限制状态增长 (p. 322).

指标表达式	统计数据	Threshold	描述
`lastCheckpointSize` > threshold	Maximum	The maximum expected checkpoint size during normal conditions.	onitors how much data is stored in state and how long it takes to take a checkpoint. If checkpoints grow or take long, the application is continuously spending time on checkpointing and has less cycles for actual processing. At some points, checkpoints may grow so or take so long that they fail. In addition to monitoring absolute values, customers should also considering monitoring the change rate with # `(lastCheckpointSize#` and # `(lastCheckpointDuration#`. If the `lastCheckpointSize` continuously increases, rising above this threshold can indicate that the application is accumulating state data. If the state data becomes too large, the application can run out of memory when recovering from a checkpoint, or recovering from a checkpoint might take too long. For troubleshooting, see 无限制状态增长 (p. 322).

指标表达式	统计数据	Threshold	描述
`heapMemoryUtilization` > threshold	Maximum	This gives a good indication of the overall resource utilization of the application and can be used for proactive scaling unless the application is I/O bound. The maximum expected `heapMemoryUtilization` size during normal conditions, with a recommended value of 90 percent.	You can use this metric to monitor the maximum memory utilization of task managers across the application. If the application reaches this threshold, you need to provision more resources. You do this by enabling automatic scaling or increasing the application parallelism. For more information about increasing resources, see 扩缩 (p. 27).
`cpuUtilization` > threshold	Maximum	This gives a good indication of the overall resource utilization of the application and can be used for proactive scaling unless the application is I/O bound. The maximum expected `cpuUtilization` size during normal conditions, with a recommended value of 80 percent.	You can use this metric to monitor the maximum CPU utilization of task managers across the application. If the application reaches this threshold, you need to provision more resources You do this by enabling automatic scaling or increasing the application parallelism. For more information about increasing resources, see 扩缩 (p. 27).
`###` > threshold	Maximum	The maximum expected `###` size during normal conditions.	You can use this metric to watch for thread leaks in task managers across the application. If this metric reaches this threshold, check your application code for threads being created without being closed.

指标表达式	统计数据	Threshold	描述
`(oldGarbageCollection## * 100) /60_000## 1 ######') >` threshold	Maximum	The maximum expected `oldGarbageCollection` time duration. We recommend setting a threshold such that typical garbage collection time is 60 percent of the specified threshold, but the correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that there is a memory leak in task managers across the application.
`#(oldGarbageCollection##)` > threshold	Maximum	The maximum expected `oldGarbageCollection##` under normal conditions. The correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that there is a memory leak in task managers across the application.
`####currentOutputWatermark-####currentInputWatermark` > threshold	Minimum	The minimum expected watermark increment under normal conditions. The correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that either the application is processing increasingly older events, or that an upstream subtask has not sent a watermark in an increasingly long time.

将自定义消息写入到 CloudWatch 日志

您可以将自定义消息写入到 Kinesis Data Analytics 应用程序的 CloudWatch 日志。您可以使用 Apache `log4j` 库或 `Simple Logging Facade for Java (SLF4J)` 库以执行该操作。

主题

- 写入 CloudWatch 使用 Log4J 日志 (p. 232)
- 写入 CloudWatch 使用 SLF4J 的日志 (p. 233)

写入 CloudWatch 使用 Log4J 日志

1. 将以下依赖项添加到应用程序的 `pom.xml` 文件中：

```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.6.1</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
```

```
    <version>2.6.1</version>
</dependency>
```

2. 包括库中的对象：

```
import org.apache.logging.log4j.Logger;
```

3. 实例化 Logger 对象并传入您的应用程序类：

```
private static final Logger log = Logger.getLogger(YourApplicationClass.class);
```

4. 使用 log.info 写入到日志。将在应用程序日志中写入大量消息。为了便于筛选自定义消息，请使用 INFO 应用程序日志级别。

```
log.info("This message will be written to the application's CloudWatch log");
```

应用程序在日志中写入一条记录，并显示类似下面的消息：

```
{
  "locationInformation":
 "com.amazonaws.services.kinesisanalytics.StreamingJob.main(StreamingJob.java:95)",
  "logger": "com.amazonaws.services.kinesisanalytics.StreamingJob",
  "message": "This message will be written to the application's CloudWatch log",
  "threadName": "Flink-DispatcherRestEndpoint-thread-2",
  "applicationARN": "arn:aws:kinesisanalytics:us-east-1:123456789012:application/test",
  "applicationVersionId": "1", "messageSchemaVersion": "1",
  "messageType": "INFO"
}
```

写入 CloudWatch 使用 SLF4J 的日志

1. 将以下依赖项添加到应用程序的 pom.xml 文件中：

```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.7</version>
    <scope>runtime</scope>
</dependency>
```

2. 包括库中的对象：

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

3. 实例化 Logger 对象并传入您的应用程序类：

```
private static final Logger log = LoggerFactory.getLogger(YourApplicationClass.class);
```

4. 使用 log.info 写入到日志。将在应用程序日志中写入大量消息。为了便于筛选自定义消息，请使用 INFO 应用程序日志级别。

```
log.info("This message will be written to the application's CloudWatch log");
```

应用程序在日志中写入一条记录，并显示类似下面的消息：

```
{
  "locationInformation":
 "com.amazonaws.services.kinesisanalytics.StreamingJob.main(StreamingJob.java:95)",
  "logger": "com.amazonaws.services.kinesisanalytics.StreamingJob",
  "message": "This message will be written to the application's CloudWatch log",
  "threadName": "Flink-DispatcherRestEndpoint-thread-2",
  "applicationARN": "arn:aws:kinesisanalytics:us-east-1:123456789012:application/test",
  "applicationVersionId": "1", "messageSchemaVersion": "1",
  "messageType": "INFO"
}
```

使用记录 Kinesis Data Analytics APIAmazon CloudTrail

Amazon Kinesis Data AnalyticsAmazon CloudTrail，提供用户、角色或角色所采取操作记录的服务Amazon 在 Kinesis Data Analytics CloudTrail 将 Kinesis Data Analytics 的所有 API 调用作为事件捕获。捕获调用包含来自 Kinesis Data Analytics 控制台的调用和对 Kinesis Data Analytics API 操作的代码调用。如果您创建了跟踪，则可以使持续交付 CloudTrail 事件到 Amazon S3 存储桶（包括 Kinesis Data Analytics 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台事件历史记录. 使用收集的信息 CloudTrail，您可以确定向 Kinesis Data Analytics 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

了解相关更多信息 CloudTrail，请参阅Amazon CloudTrail用户指南.

Kinesis Data Analytics CloudTrail

CloudTrail 已在您的Amazon在您创建账户时。当 Kinesis Data Analytics 中发生活动时，该活动将记录在 CloudTrail 活动与其他Amazon服务事件事件历史记录. 您可以在 Amazon 账户中查看、搜索和下载最新事件。有关更多信息，请参阅 。使用查看事件 CloudTrail 事件历史记录.

要持续记录在Amazon账户（包括 Kinesis Data Analytics 的事件）创建跟踪。一个踪迹启用 CloudTrail 将日志文件传送至 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 Amazon 区域。此跟踪记录在 Amazon 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service（Amazon S3）存储桶。此外，您还可以配置其他Amazon服务，进一步分析在中收集的事件数据并采取措施 CloudTrail 日志。有关更多信息，请参阅下列内容：

- 创建跟踪记录概述
- CloudTrail 支持的服务和集成
- 为 配置 Amazon SNS 通知 CloudTrail
- 接收 CloudTrail 多个区域中的日志文件和接收 CloudTrail 多个账户中的日志文件

所有 Kinesis Data Analytics 操作均由记录 CloudTrail 并记录在Kinesis Data Analytics. 例如，对`CreateApplication`和 `UpdateApplication`操作的调用在 CloudTrail 日志文件。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 Amazon Identity and Access Management（IAM）用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它 Amazon 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 Kinesis Data Analytics 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。 CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

以下示例显示了一个 CloudTrail 说明的日志条目AddApplicationCloudWatchLoggingOption和DescribeApplication行动。

```
{
    "Records": [
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::012345678910:user/Alice",
                "accountId": "012345678910",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
            },
            "eventTime": "2019-03-07T01:19:47Z",
            "eventSource": "kinesisanalytics.amazonaws.com",
            "eventName": "AddApplicationCloudWatchLoggingOption",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
            "requestParameters": {
                "applicationName": "cloudtrail-test",
                "currentApplicationVersionId": 1,
                "cloudWatchLoggingOption": {
                    "logStreamARN": "arn:aws:logs:us-east-1:012345678910:log-
group:cloudtrail-test:log-stream:flink-cloudwatch"
                }
            },
            "responseElements": {
                "cloudWatchLoggingOptionDescriptions": [
                    {
                        "cloudWatchLoggingOptionId": "2.1",
                        "logStreamARN": "arn:aws:logs:us-east-1:012345678910:log-
group:cloudtrail-test:log-stream:flink-cloudwatch"
                    }
                ],
                "applicationVersionId": 2,
                "applicationARN": "arn:aws:kinesisanalytics:us-
east-1:012345678910:application/cloudtrail-test"
            },
            "requestID": "18dfb315-4077-11e9-afd3-67f7af21e34f",
            "eventID": "d3c9e467-db1d-4cab-a628-c21258385124",
            "eventType": "AwsApiCall",
            "apiVersion": "2018-05-23",
            "recipientAccountId": "012345678910"
        },
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::012345678910:user/Alice",
                "accountId": "012345678910",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
```

```
            },
            "eventTime": "2019-03-12T02:40:48Z",
            "eventSource": "kinesisanalytics.amazonaws.com",
            "eventName": "DescribeApplication",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
            "requestParameters": {
                "applicationName": "sample-app"
            },
            "responseElements": null,
            "requestID": "3e82dc3e-4470-11e9-9d01-e789c4e9a3ca",
            "eventID": "90ffe8e4-9e47-48c9-84e1-4f2d427d98a5",
            "eventType": "AwsApiCall",
            "apiVersion": "2018-05-23",
            "recipientAccountId": "012345678910"
        }
    ]
}
```

为 Apache Flink 调整 Amazon Kinesis Data Analytics 中的性能

本主题介绍了监控和提高 Kinesis Data Analytics 应用程序性能的技术。

主题

排查性能方

本节包含一个症状列表，您可以检查这些症状以诊断和修复性能问题。

如果您的数据源是 Kinesis 流，则性能问题通常表现为高或不断增加`MillisBehindLatest`指标。对于其他来源，您可以检查表示从源读取滞后的类似指标。

数据路径

在调查应用程序的性能问题时，请考虑数据所采用的整个路径。如果设计或配置不正确，以下应用程序组件可能会成为性能瓶颈并造成背压：

- 数据源和目标：确保应用程序与之交互的外部资源为应用程序将体验的吞吐量配置了属性。
- 状态数据：确保您的应用程序不会频繁与州商店进行交互。

  您可以优化应用程序正在使用的序列化器。默认的 Kryo 序列化器可以处理任何可序列化的类型，但是如果应用程序仅以 POJO 类型存储数据，则可以使用更高性能的序列化器。有关 Apache Flink 序列化器的信息，请参阅数据类型和序列化中的Apache Flink 文档.
- 运算符：确保运营商实施的业务逻辑不会太复杂，或者您在处理每条记录时没有创建或使用资源。还要确保您的应用程序不会过于频繁地创建滑动或翻滚窗口。

性能故障排除解

本节包含性能问题的潜在解决方案。

主题

CloudWatch 监控级别

验证 CloudWatch 监控级别没有设置为过于冗长设置。

这些区域有：`Debug`监视日志级别设置会产生大量流量，这可能会造成背压。只有在积极调查应用程序的问题时才能使用它。

如果你的应用程序有很高`Parallelism`设置，请使用 `Parallelism`监控指标级别同样会产生大量流量，从而导致背压。仅在以下情况下使用此指标级`Parallelism`因为你的应用程序不足，或者在调查应用程序的问题时。

有关更多信息，请参阅 应用程序监控级别 (p. 210)。

应用程序 CPU 指标

检查应用程序的`CPU`指标。如果此指标超过 75%，则可以通过启用 auto 扩展来允许应用程序为自己分配更多资源。

如果启用了 auto 扩展，如果 CPU 使用率在 15 分钟内超过 75%，则应用程序会分配更多资源。有关扩展的更多信息，请参阅正确管理扩展 (p. 239)下面的部分，以及扩缩 (p. 27).

> Note
>
> 应用程序只会根据 CPU 使用情况自动扩展。应用程序不会 auto 扩展以响应其他系统指标，例如`heapMemoryUtilization`. 如果您的应用程序对其他指标的使用率很高，请手动提高应用程序的并行度。

应用程序并行度

增加应用程序的并行度。您可以使用`ParallelismConfigurationUpdate`的参数UpdateApplicationaction.

应用程序的最大 KPU 默认为 32 个，可以请求增加限制以增加该数值。

还必须根据每个运营商的工作负载为其分配并行性，而不仅仅是增加应用程序并行度。请参阅运算符并行度 (p. 238)以下。

应用程序记

检查应用程序是否为处理的每个记录写入一个条目。在应用程序具有较高的吞吐量时，为每个记录写入一个日志条目将导致严重的数据处理瓶颈。要检查这种情况，请查询日志以查找应用程序为它处理的每个记录写入的日志条目。有关读应用程序日志的更多信息，请参阅the section called “分析日志” (p. 211).

运算符并行度

验证是否在工作进程中均匀分配应用程序的工作负载。

有关调整应用程序操作员工作负载的信息，请参阅运营商扩展 (p. 239).

应用逻辑

检查应用程序逻辑以查找效率低下或性能不佳的操作，例如，访问外部依赖项（如数据库或 Web 服务），访问应用程序状态，等等。如果外部依赖关系不是性能或不可靠地访问，也可能会阻碍性能，这可能会导致外部依赖关系退出`HTTP 500`错误消息。

如果应用程序使用外部依赖项以丰富或以其他方式处理传入数据，请考虑改用异步 IO。有关更多信息，请参阅 。异步 I/O中的Apache Flink 文档.

应用程序存

检查应用程序是否有资源泄漏。如果你的应用程序没有正确处置线程或内存，你可能会看到`MillisBehindLatest`、`CheckpointSize`, 和`CheckpointDuration`指标激增或逐渐增加。这种情况也可能导致任务管理器或作业管理器失败。

性能最佳实践

本节介绍设计应用程序以提高性能的特殊注意事项。

正确管理扩展

本节包含有关管理应用程序级和操作员级扩展的信息。

本节包含以下主题：

- 正确管理应用程序扩 (p. 239)
- 正确管理运营商扩展 (p. 239)

正确管理应用程序扩

您可以使用自动缩放来处理应用程序活动中的意外峰值。如果满足以下条件，则应用程序的 KPU 将自动增加：

- 已为应用程序启用自动缩放功能。
- 在 15 分钟内，CPU 使用率保持在 75% 以上。

如果启用了自动缩放，但 CPU 使用率没有保持在此阈值，则应用程序将不会扩展 KPU。如果您遇到不符合此阈值的 CPU 使用率飙升，或者在不同的使用量指标（例如）`heapMemoryUtilization`，手动增加扩展以允许您的应用程序处理活动峰值。

> Note
>
> 如果应用程序通过 auto Scaling 自动添加了更多资源，则应用程序将在处于不活动状态一段时间后释放新资源。缩减资源将暂时影响性能。

有关扩展的更多信息，请参阅扩缩 (p. 27).

正确管理运营商扩展

您可以通过验证应用程序的工作负载在工作进程之间均匀分配，以及应用程序中的操作员拥有稳定性和高性能所需的系统资源，可以提高应用程序的性能。

您可以使用`parallelism`设置。如果您没有为运算符设置并行度，它将使用应用程序级别的并行度设置。使用应用程序级并行度设置的操作员可能会使用应用程序可用的所有系统资源，从而使应用程序不稳定。

为了最好地确定每个运算符的并行度，请考虑运营商与应用程序中其他运算符相比的相对资源需求。将资源密集型运算符设置为比资源密集型较低的运算符更高的并行度设置。

应用程序的总运算符并行度是应用程序中所有运算符的并行度之和。您可以通过确定应用程序与应用程序可用的总任务槽之间的最佳比率来调整应用程序的总运算符并行度。总操作员并行度与任务槽之间的典型稳定比例为 4:1，也就是说，应用程序每四个可用的操作员子任务都有一个任务槽可用。拥有更多资源密集型运营商的应用程序可能需要 3:1 或 2:1 的比率，而资源密集型运营商的应用程序可能需要比例为 10:1 的稳定。

你可以使用以下方法为操作员设置比率运行时属性 (p. 18)，因此您可以在不编译和上传应用程序代码的情况下调整运算符的并行度。

以下代码示例说明了如何将运算符并行度设置为当前应用程序并行度的可调比率：

```
Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
operatorParallelism =
    StreamExecutionEnvironment.getParallelism() /
    Integer.getInteger(

 applicationProperties.get("OperatorProperties").getProperty("MyOperatorParallelismRatio")
    );
```

有关子任务、任务槽和其他应用程序资源的信息，请参阅应用程序资 (p. 7).

要控制如何在应用程序的工作进程中分配工作负载，请使用`Parallelism`设置和`KeyBy`分区方法。有关更多信息，请参阅中的以下主题：Apache Flink 文档：

- 并行执行
- DataStream 转换

监控外部依赖资源的使用

如果目标中存在性能瓶颈（例如 Kinesis Streams、Kinesis Data Firehose、DynamoDB 或 OpenSearch 服务），你的应用程序将遇到背压。验证外部依赖项是否已针对应用程序吞吐量正确配置。

> Note
>
> 其他服务的故障可能会导致应用程序出现故障。如果你看到应用程序中的失败，请检查 CloudWatch 目标服务的日志是否出现故障。

在本地运行 Apache Flink 应用程序

要解决内存问题，您可以在本地 Flink 安装中运行应用程序。这将允许您访问在 Kinesis Data Analytics 中运行应用程序时不可用的调试工具，例如堆栈跟踪和堆转储。

有关创建本地 Flink 安装的信息，请参阅本地安装教程中的Apache Flink 文档.

监控性能

本节介绍用于监控应用程序性能的工具。

使用监控性能 CloudWatch 指标

您可以使用监控应用程序的资源使用情况、吞吐量、检查点和停机时间 CloudWatch 指标。有关使用的信息 CloudWatch 使用 Kinesis Data Analytics 应用程序指标，请参阅指标与维度 (p. 214).

使用监控性能 CloudWatch 日志和警报

您可以使用监控可能导致性能问题的错误情况 CloudWatch 日志。

当 Apache Flink 作业状态从`RUNNING`状态为`FAILED`状态。

你用 CloudWatch 警报以创建性能问题的通知，例如资源使用情况或超过安全阈值的检查点指标，或应用程序状态意外更改。

有关创建的信息 CloudWatch Kinesis Data Analytics 应用程序的警报，请参阅告警 (p. 226).

Kinesis Data Analytics 针对 Apache Flink 和 Studio 笔记本电脑配额

使用 Amazon Kinesis Data Analytics 针对 Apache Flink 时，请注意以下配额：

- 您可以在您的账户中为每个区域创建最多 50 个 Kinesis Data Analytics 应用程序。您可以创建案例，通过服务配额增加表来申请其他应用程序。有关更多信息，请参阅 。Amazon Web Services SupportCenter.

  有关支持 Kinesis Data Analytics 的区域的列表，请参阅Kinesis Data Analytics 区域和终端.

- 默认情况下，Kinesis 处理单元 (KPU) 数量限制为 32 个。有关如何请求增加该配额的说明，请参阅请求提高配额在Service Quotas. 确保指定需要应用新的 KPU 限制的应用程序前缀。

  使用 Kinesis Data Analytics，Amazon根据分配的资源进行付费，而不是根据应用程序使用的资源。将根据用于运行流处理应用程序的最大 KPU 数按小时费率对您收费。一个 KPU 可为您提供 1 个 vCPU 和 4 GiB 内存。对于每个 KPU，该服务还预置 50 GiB 的运行应用程序存储。

- 您可以创建最多 1000 个 Kinesis Data Analytics快照 (p. 24)每个应用程序。

- 您可以为每个应用程序分配最多 50 个标签。

- 应用程序 JAR 文件的最大大小为 512 MiB。如果超过该配额，应用程序将无法启动。

对于 Studio 笔记本，以下配额将适用。要请求提高配额，创建支持案例.

- `websocketMessageSize`= 5 MiB
- `noteSize`= 5 MiB
- `noteCount`= 1000
- `Max cumulative UDF size`= 100 MiB
- `Max cumulative dependency jar size`= 300 MiB

针对 Apache Flink 维护的 Kinesis Data Analytics

Kinesis Data Analytics 通过操作系统和容器映像安全更新定期修补应用程序，以保持合规性并满足Amazon 安全目标。下表列出了 Kinesis Data Analytics 执行此类维护的默认时间窗口。您的应用程序的维护可能会在与您所在地区对应的时间窗口内的任何时间进行。在此维护过程中，您的应用程序可能会遇到 10 到 30 秒的停机时间。但是，实际停机时间持续时间取决于应用程序状态。有关如何最大限度地减少此停机时间的影响的信息，请参阅the section called “容错：检查点和保存点” (p. 246).

要更改 Kinesis Data Analytics 对应用程序执行维护的时间窗口，请使用UpdateApplicationMaintenanceConfigurationAPI。

区域	维护时段
美国东部（弗吉尼亚北部）	03:00–11:00 UTC
Amazon GovCloud（美国东部）	03:00–11:00 UTC
美国东部（俄亥俄）	03:00–11:00 UTC
美国西部（加利福尼亚北部）	06:00–14:00 UTC
美国西部（俄勒冈）	06:00–14:00 UTC
Amazon GovCloud（美国西部）	06:00–14:00 UTC
亚太地区（香港）	13:00–21:00 UTC
亚太地区（孟买）	16:30–00:30 UTC
亚太地区（首尔）	13:00–21:00 UTC
亚太地区（新加坡）	14:00–22:00 UTC
亚太地区（悉尼）	12:00–20:00 UTC
亚太地区（东京）	13:00–21:00 UTC
加拿大（中部）	03:00–11:00 UTC
中国（北京）	13:00–21:00 UTC
中国（宁夏）	13:00–21:00 UTC
欧洲（法兰克福）	06:00–14:00 UTC
欧洲（爱尔兰）	22:00–06:00 UTC
欧洲（伦敦）	22:00–06:00 UTC
欧洲（米兰）	21:00–05:00 UTC
欧洲（巴黎）	23:00–07:00 UTC
欧洲（斯德哥尔摩）	23:00–07:00 UTC

区域	维护时段
Middle East (Bahrain)	13:00–21:00 UTC
南美洲（圣保罗）	世界标准时间 19:00-03:00

准备就绪

这是在 Kinesis Data Analytics 上运行生产应用程序的重要方面的集合。这不是一个详尽的清单，而是在将应用程序投入生产之前你应该注意的最低限度的清单。

负载测试应用程序

应用程序的某些问题仅在负载过重的情况下才会显现出来。我们已经看到，应用程序看起来很健康，运行事件大大放大了应用程序的负担。这可能完全独立于应用程序本身：如果数据源或数据接收器在几个小时内不可用，Flink 应用程序将无法继续运行。一旦该问题得到解决，就会积累大量未处理的数据，这可能会完全耗尽可用资源。然后，负载可能会放大以前从未出现过的错误或性能问题。

因此，必须为生产应用程序运行适当的负载测试。在这些负载测试中应回答的问题包括：

- 应用程序在持续的高负载下是否稳定？
- 在峰值负载下，应用程序还能使用保存点吗？
- 处理来自 1 小时的积压需要多长时间？ 24 小时需要多长时间（取决于流中数据的最大保留时间）？
- 当应用程序扩展时，应用程序的吞吐量是否会增加？

使用数据流时，可以通过在流中生成一段时间来模拟这些场景。然后启动应用程序，让它从一开始就使用数据，例如，使用起始位置`TRIM_HORIZON`对于 Kinesis 数据流。

为 DAG 中的每个运算符设置显式的最大并行度

这应该是你的函数`total parallelism`用于该应用程序。例如，如果您的总并行度为 10，则可以将单个运算符设置为总并行度的一半，例如：

```
.setParallelism(MAX_PARALLELISM/2)
```

哪里`MAX_PARALLELISM`是一个等于的结果的变量`env.getParallelism()`.

为所有运算符设置 UUID

在 Flink 将保存点映射回单个运算符的操作中使用 UUID。为每个运算符设置特定的 UUID 可为要还原的保存点进程提供稳定的映射。

```
.map(...).uid("my-map-function")
```

有关更多信息，请参阅 。生产准备就绪核对清单.

Apache Flink Kinesis Data Analytics 的最佳实践

本节包含有关开发稳定的高性能的 Amazon Kinesis Data Analytics 应用程序的信息和建议。

主题

容错：检查点和保存点

使用检查点和保存点在 Kinesis Data Analytics for Apache Flink 应用程序中实施容错功能。在开发和维护应用程序时，请牢记以下几点：

- 我们建议您为应用程序启用检查点。检查点可以为应用程序在计划维护期间以及由于服务问题、应用程序依赖项故障和其他问题而导致意外故障时提供容错功能。有关计划维护的信息，请参阅维护 (p. 243).
- SetApplicationSnapshotConfiguration::SnapshotsEnabled到`false`在应用程序开发或故障排除期间。在每次应用程序停止期间，将会创建一个快照；如果应用程序处于不正常状态或性能不佳，则可能会出现问题。在应用程序处于生产状态并保持稳定后，将 `SnapshotsEnabled` 设置为 `true`。

  Note

  我们建议应用程序每天创建多次快照，以使用正确的状态数据正确地重新启动。快照的正确频率取决于应用程序的业务逻辑。频繁拍摄快照允许您恢复较新的数据，但会增加成本并需要更多的系统资源。

  有关监控应用程序停机时间的信息，请参阅指标与维度 (p. 214)。

有关实施容错功能的更多信息，请参阅容错能力 (p. 21)。

性能和并行度

应用程序可以调整应用程序并行度并避免性能陷阱，从而进行扩展以满足任何吞吐量级别要求。在开发和维护应用程序时，请牢记以下几点：

- 验证是否充分预置了所有应用程序源和接收器，而不会受到限制。如果源和接收器是其他 Amazon 服务，请使用 CloudWatch 监控这些服务。

- 对于并行度较高的应用程序，请检查是否将较高的并行度应用于应用程序中的所有操作符。默认情况下，Apache Flink 为应用程序图中的所有操作符应用相同的应用程序并行度。这可能会导致在源或接收器上出现预置问题，或者出现操作符数据处理瓶颈。您可以使用 setParallelism 更改代码中的每个操作符的并行度。
- 了解应用程序中的操作符的并行度设置的含义。如果更改操作符的并行度，您可能无法从操作符并行度与当前设置不兼容时创建的快照中还原应用程序。有关设置运算符并行的更多信息，请参阅显式设置运算符的最大并行度.

有关实施扩展的更多信息，请参阅扩缩 (p. 27)。

日志记录

您可以使用应用程序的性能和错误情况监视应用程序的性能和错误情况 CloudWatch 日志。为应用程序配置日志记录时，请牢记以下几点：

- 启用 CloudWatch 记录应用程序以便可以调试任何运行时问题。
- 不要为应用程序中处理的每条记录创建一个日志条目。这会在处理期间出现严重瓶颈，并且可能会导致数据处理反向压力。
- Create CloudWatch 警报会在应用程序未正常运行时通知您。有关更多信息，请参阅 。告警 (p. 226)

有关实施日志记录的更多信息，请参阅日志记录和监控 (p. 204)。

编码

您可以使用建议的编程做法以提高应用程序性能和稳定性。在编写应用程序代码时，请牢记以下几点：

- 不要在应用程序代码（应用程序的 `main` 方法或用户定义的函数）中使用 `system.exit()`。如果要从代码中关闭应用程序，请引发一个从 `Exception` 或 `RuntimeException` 派生的异常，其中包含有关应用程序出现的错误的消息。

  请注意下面有关该服务如何处理此类异常的信息：
  - 如果你的应用程序抛出异常`main`方法时，该服务会将其封装在`ProgramInvocationException`当应用程序转换到`RUNNING`status，则作业经理将无法提交作业。
  - 如果异常是从用户定义的函数中引发的，作业管理器使作业失败并重新启动，并将异常详细信息写入到异常日志中。
- 请考虑为应用程序 JAR 文件及其包含的依赖项填充阴影。如果应用程序和 Apache Flink 运行时的程序包名称存在潜在的冲突，则建议填充阴影。如果发生冲突，则应用程序日志可能包含 `java.util.concurrent.ExecutionException` 类型的异常。有关为应用程序 JAR 文件填充阴影的更多信息，请参阅 Apache Maven Shade 插件。

管理凭证

您不应将任何长期凭证纳入生产（或任何其他）应用程序。长期凭证很可能被签入版本控制系统，很容易丢失。相反，您可以将角色关联到 Kinesis Data Analytics 应用程序，并向该角色授予权限。然后，正在运行的 Flink 应用程序可以从环境中获取具有相应权限的临时证书。如果未与 IAM 进行本机集成的服务需要身份验证，例如需要用户名和密码进行身份验证的数据库，则应考虑将密钥存储在AmazonSecrets Manager.

多点Amazon本机服务支持身份验证：

- Kinesis Data StreamsProcessTaxiStream.开发工具包

- Amazon MSK —https://github.com/aws/aws-msk-iam-auth/#using-the-amazon-msk-library-for-iam-authentication
- Amazon Elasticsearch ServiceAmazonElasticsearchSink.开发工具包
- Amazon S3 — 开箱即用 Kinesis Data Analytics

从分片/分区较少的源读取

从 Apache Kafka 或 Kinesis 数据流读取数据时，流的并行度（即 Kafka 的分区数量和 Kinesis 的分片数量）与应用程序的并行度之间可能存在不匹配。使用朴素的设计，应用程序的并行性无法扩展到流的并行度之外：源运算符的每个子任务只能读取 1 个或多个分片/分区。这意味着对于只有 2 个分片的流和并行度为 8 的应用程序，实际上只有两个子任务正在从流中消耗，6 个子任务仍处于空闲状态。这可能会极大地限制应用程序的吞吐量，特别是如果反序列化代价高昂且由源执行（这是默认设置）。

要减轻这种影响，您可以缩放流。但这可能并不总是可取或可能的。或者，您可以重构源代码，使其不进行任何序列化，而只是传递`byte[]`. 然后您可以再平衡将数据均匀分布到所有任务中，然后在那里反序列化数据。通过这种方式，您可以利用所有子任务进行反序列化，而这种可能昂贵的操作不再受流的分片/分区数量的限制。

Studio 笔记本更新间隔

如果更改段落结果刷新间隔，请将其设置为至少 1000 毫秒的值。

Studio 笔记本最佳性能

我们使用以下语句进行了测试，并在以下情况下获得了最佳性能`events-per-second`乘以`number-of-keys`低于 25,000,000。这是为了`events-per-second`低于 15 万。

```
SELECT key, sum(value) FROM key-values GROUP BY key
```

水印策略和空闲分片如何影响时间窗口

从 Apache Kafka 和 Kinesis Data Streams 中读取事件时，源可以根据流的属性设置事件时间。对于 Kinesis，事件时间等于事件的大致到达时间。但是，在事件源位置设置事件时间不足以让 Flink 应用程序使用事件时间。源还必须生成水印，以便将有关事件时间的信息从源传播到所有其他运算符。这些区域有：Flink 文档很好地概述了该过程的工作原理。

默认情况下，从 Kinesis 读取的事件的时间戳设置为由 Kinesis 确定的近似到达时间。事件时间在应用程序中起作用的另一个先决条件是水印策略。

```
WatermarkStrategy<String> s = WatermarkStrategy
    .<String>forMonotonousTimestamps()
    .withIdleness(Duration.ofSeconds(...));
```

然后将水印策略应用于`DataStream`用`assignTimestampsAndWatermarks`方法。有一些有用的内置策略：

- `forMonotonousTimestamps()`将只使用事件时间（近似到达时间）并定期发出最大值作为水印（对于每个特定的子任务）

- `forBoundedOutOfOrderness(Duration.ofSeconds(...))`与之前的策略类似，但将使用事件时间-持续时间来生成水印。

这行得通，但有几个注意事项需要注意。水印在子任务级别生成，并流过运算符图。

从Flink 文档：

源函数的每个parallel 子任务通常独立生成其水印。这些水印定义了该特定parallel 源的事件时间。

当水印流过流媒体程序时，它们会提前到达运营商的活动时间。每当运算符提前其事件时间时，它都会为其后继运算符生成一个新的下游水印。

有些运算符消耗多个输入流；例如，联合运算符，或跟随 KeyBy (...) 或分区 (...) 函数的运算符。此类运算符的当前事件时间是其输入流事件时间的最小值。当其输入流更新其事件时间时，运算符也会更新。

这意味着，如果源子任务正在使用空闲分片，则下游操作员不会从该子任务收到新的水印，因此所有使用时间窗的下游运算符都不会处理停顿。为避免这种情况，客户可以添加`withIdleness`水印策略的选项。使用该选项，运算符在计算操作员的事件时间时会从空闲的上游子任务中排除水印。因此，空闲子任务不再阻碍下游运算符的事件时间的提高。

但是，如果没有子任务正在读取任何事件，即流中没有事件，则带有内置水印策略的 idleness 选项不会提前事件时间。这对于从流中读取一组有限事件的测试用例尤其明显。由于读取最后一个事件后事件时间不会提前，因此最后一个窗口（包含最后一个事件）将永远不会关闭。

摘要

- 这`withIdleness`如果分片处于空闲状态，设置不会生成新的水印，它只会从下游运算符的最小水位线计算中排除空闲子任务发送的最后一个水印
- 使用内置的水印策略，最后一个打开的窗口将永远不会关闭（除非发送推进水印的新事件，但这会创建一个新窗口，然后保持打开状态）
- 即使时间由 Kinesis 流设置，如果一个分片的消耗速度快于其他分片（例如，在应用初始化期间或使用`TRIM_HORIZON`其中所有现有分片都被parallel 消耗，忽略了它们的父/子关系）
- 这`withIdleness`水印策略的设置似乎弃用了空闲分片的 Kinesis 源特定设置`(ConsumerConfigConstants.SHARD_IDLE_INTERVAL_MILLIS`

示例

以下应用程序正在从流中读取数据，并根据事件时间创建会话窗口。

```
Properties consumerConfig = new Properties();
consumerConfig.put(AWSConfigConstants.AWS_REGION, "eu-west-1");
consumerConfig.put(ConsumerConfigConstants.STREAM_INITIAL_POSITION, "TRIM_HORIZON");

FlinkKinesisConsumer<String> consumer = new FlinkKinesisConsumer<>("...", new
 SimpleStringSchema(), consumerConfig);

WatermarkStrategy<String> s = WatermarkStrategy
    .<String>forMonotonousTimestamps()
    .withIdleness(Duration.ofSeconds(15));

env.addSource(consumer)
    .assignTimestampsAndWatermarks(s)
    .map(new MapFunction<String, Long>() {
        @Override
        public Long map(String s) throws Exception {
            return Long.parseLong(s);
```

```
        }
    })
    .keyBy(l -> 0l)
    .window(EventTimeSessionWindows.withGap(Time.seconds(10)))
    .process(new ProcessWindowFunction<Long, Object, Long, TimeWindow>() {
        @Override
        public void process(Long aLong, ProcessWindowFunction<Long, Object, Long,
 TimeWindow>.Context context, Iterable<Long>iterable, Collector<Object> collector) throws
 Exception {
            long count = StreamSupport.stream(iterable.spliterator(), false).count();
            long timestamp = context.currentWatermark();

            System.out.print("XXXXXXXXXXXXXX Window with " + count + " events");
            System.out.println("; Watermark: " + timestamp + ", " +
 Instant.ofEpochMilli(timestamp));


            for (Long l : iterable) {
                System.out.println(l);
            }
        }
    });
```

在以下示例中，8 个事件被写入 16 个分片流（前 2 个事件和最后一个事件恰好落在同一个分片中）。

```
$ aws kinesis put-record --stream-name hp-16 --partition-key 1 --data MQ==
$ aws kinesis put-record --stream-name hp-16 --partition-key 2 --data Mg==
$ aws kinesis put-record --stream-name hp-16 --partition-key 3 --data Mw==
$ date

{
    "ShardId": "shardId-000000000012",
    "SequenceNumber": "49627894338614655560500811028721934184977530127978070210"
}
{
    "ShardId": "shardId-000000000012",
    "SequenceNumber": "49627894338614655560500811028795678659974022576354623682"
}
{
    "ShardId": "shardId-000000000014",
    "SequenceNumber": "49627894338659257050897872275134360684221592378842022114"
}
Wed Mar 23 11:19:57 CET 2022

$ sleep 10
$ aws kinesis put-record --stream-name hp-16 --partition-key 4 --data NA==
$ aws kinesis put-record --stream-name hp-16 --partition-key 5 --data NQ==
$ date

{
    "ShardId": "shardId-000000000010",
    "SequenceNumber": "49627894338570054070103749783042116732419934393936642210"
}
{
    "ShardId": "shardId-000000000014",
    "SequenceNumber": "49627894338659257050897872275659034489934342334017700066"
}
Wed Mar 23 11:20:10 CET 2022

$ sleep 10
$ aws kinesis put-record --stream-name hp-16 --partition-key 6 --data Ng==
$ date

{
    "ShardId": "shardId-000000000001",
```

```
    "SequenceNumber": "496278943383693473633169741738869883454670353653752135 86"
}
Wed Mar 23 11:20:22 CET 2022

$ sleep 10
$ aws kinesis put-record --stream-name hp-16 --partition-key 7 --data Nw==
$ date

{
    "ShardId": "shardId-000000000008",
    "SequenceNumber": "496278943385254525797066885358789472991951893497255036 18"
}
Wed Mar 23 11:20:34 CET 2022

$ sleep 60
$ aws kinesis put-record --stream-name hp-16 --partition-key 8 --data OA==
$ date

{
    "ShardId": "shardId-000000000012",
    "SequenceNumber": "496278943386146555605008110296008232558373719289007966 10"
}
Wed Mar 23 11:21:27 CET 2022
```

此输入应导致 5 个会话窗口：事件 1,2,3；事件 4,5；事件 6；事件 7；事件 8。但是，该程序仅生成前 4 个窗口。

```
11:59:21,529 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 5 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000006,HashKeyRange: {StartingHashKey:
 127605887595351923798765477786913079296,EndingHashKey:
 148873535527910577765226390751398592511},SequenceNumberRange: {StartingSequenceNumber:
 496278943384808510893096272895245492392926255883957044 18,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 5 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000006,HashKeyRange: {StartingHashKey:
 127605887595351923798765477786913079296,EndingHashKey:
 148873535527910577765226390751398592511},SequenceNumberRange: {StartingSequenceNumber:
 496278943384808510893096272895245492392926255883957044 18,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,530 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 6 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000007,HashKeyRange: {StartingHashKey:
 148873535527910577765226390751398592512,EndingHashKey:
 170141183460469231731687303715884105727},SequenceNumberRange: {StartingSequenceNumber:
 496278943385031518345081579126660849575652739499016848 50,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 6 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000010,HashKeyRange: {StartingHashKey:
 212676479325586539664609129644855132160,EndingHashKey:
 233944127258145193631070042609340645375},SequenceNumberRange: {StartingSequenceNumber:
 496278943385700540701037497820906921123832190344196261 46,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 6 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000007,HashKeyRange: {StartingHashKey:
 148873535527910577765226390751398592512,EndingHashKey:
 170141183460469231731687303715884105727},SequenceNumberRange: {StartingSequenceNumber:
 496278943385031518345081579126660849575652739499016848 50,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
```

```
11:59:21,531 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 4 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000005,HashKeyRange: {StartingHashKey:
 106338239662793269832304564822427566080,EndingHashKey:
 127605887595351923798765477786913079295},SequenceNumberRange: {StartingSequenceNumber:
 496278943384585503441110966663830135210199772268897239 86,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 4 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000005,HashKeyRange: {StartingHashKey:
 106338239662793269832304564822427566080,EndingHashKey:
 127605887595351923798765477786913079295},SequenceNumberRange: {StartingSequenceNumber:
 496278943384585503441110966663830135210199772268897239 86,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 3 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000004,HashKeyRange: {StartingHashKey:
 850705917302346158658436518579420528 64,EndingHashKey:
 106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
 496278943384362495989125660432414778027473288653837435 54,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 2 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000003,HashKeyRange: {StartingHashKey:
 638029437976759618993827388934565396 48,EndingHashKey:
 850705917302346158658436518579420528 63},SequenceNumberRange: {StartingSequenceNumber:
 496278943384139488537140354200999420844746805038777631 22,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 3 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000015,HashKeyRange: {StartingHashKey:
 319014718988379809496913694467282698 240,EndingHashKey:
 340282366920938463463374607431768211455},SequenceNumberRange: {StartingSequenceNumber:
 496278943386815577960964028977983707037464608419495283 06,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 2 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000014,HashKeyRange: {StartingHashKey:
 297747071055821155530452781502797185024,EndingHashKey:
 319014718988379809496913694467282698239},SequenceNumberRange: {StartingSequenceNumber:
 496278943386592570508978722746568349854738124804435478 74,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 3 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000004,HashKeyRange: {StartingHashKey:
 850705917302346158658436518579420528 64,EndingHashKey:
 106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
 496278943384362495989125660432414778027473288653837435 54,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 2 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000003,HashKeyRange: {StartingHashKey:
 638029437976759618993827388934565396 48,EndingHashKey:
 850705917302346158658436518579420528 63},SequenceNumberRange: {StartingSequenceNumber:
 496278943384139488537140354200999420844746805038777631 22,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey:
 212676479325586539664609129644855132 16,EndingHashKey:
 425352958651173079329218259289710264 31},SequenceNumberRange: {StartingSequenceNumber:
 496278943383693473633169741738168706479293837808658022 58,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
```

```
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000009,HashKeyRange: {StartingHashKey:
 191408831393027885698148216680369618944,EndingHashKey:
 212676479325586539664609129644855132159},SequenceNumberRange: {StartingSequenceNumber:
 496278943385477533249052191589491563941105706729136457l4,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
 212676479325586539664609129644855132l5},SequenceNumberRange: {StartingSequenceNumber:
 496278943383470466181184435506753349296567354193598218261,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000012,HashKeyRange: {StartingHashKey:
 255211775190703847597530955573826158592,EndingHashKey:
 276479423123262501563991868538311671807},SequenceNumberRange: {StartingSequenceNumber:
 496278943386146555605008110283737635489285157574315870l0,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000008,HashKeyRange: {StartingHashKey:
 170141183460469231731687303715884105728,EndingHashKey:
 191408831393027885698148216680369618943},SequenceNumberRange: {StartingSequenceNumber:
 496278943385254525797066885358076206758379223114076652820,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey:
 212676479325586539664609129644855132l6,EndingHashKey:
 425352958651173079329218259289710264310},SequenceNumberRange: {StartingSequenceNumber:
 496278943383693473633169741738168706479293837808658022580,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000011,HashKeyRange: {StartingHashKey:
 233944127258145193631070042609340645376,EndingHashKey:
 255211775190703847597530955573826158591},SequenceNumberRange: {StartingSequenceNumber:
 496278943385923548153022804052322278306558673959256065781,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
 212676479325586539664609129644855132l5},SequenceNumberRange: {StartingSequenceNumber:
 496278943383470466181184435506753349296567354193598218261,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,568 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000002,HashKeyRange: {StartingHashKey:
 425352958651173079329218259289710264320,EndingHashKey:
 638029437976759618993827388934565396471},SequenceNumberRange: {StartingSequenceNumber:
 496278943383916481085155047969584063662020321423717826901,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,568 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000013,HashKeyRange: {StartingHashKey:
 276479423123262501563991868538311671808,EndingHashKey:
 297747071055821155530452781502797185023},SequenceNumberRange: {StartingSequenceNumber:
 496278943386369563056993416515152992672011641189375674421,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,568 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 1 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
```

```
 shard='{ShardId: shardId-000000000002,HashKeyRange: {StartingHashKey:
 42535295865117307932921825928971026432,EndingHashKey:
 63802943797675961899382738893456539647},SequenceNumberRange: {StartingSequenceNumber:
 49627894338391648108515504796958406366202032142371782690,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:23,209 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000009,HashKeyRange: {StartingHashKey:
 191408831393027885698148216680369618944,EndingHashKey:
 212676479325586539664609129644855132159},SequenceNumberRange: {StartingSequenceNumber:
 49627894338547753324905219158949156394110570672913645714,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,244 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 6 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000010,HashKeyRange: {StartingHashKey:
 212676479325586539664609129644855132160,EndingHashKey:
 233944127258145193631070042609340645375},SequenceNumberRange: {StartingSequenceNumber:
 49627894338570054070103749782090692112383219034419626146,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
event: 6; timestamp: 1648030822428, 2022-03-23T10:20:22.428Z
11:59:23,377 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 3 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000015,HashKeyRange: {StartingHashKey:
 319014718988379809496913694467282698240,EndingHashKey:
 340282366920938463463374607431768211455},SequenceNumberRange: {StartingSequenceNumber:
 49627894338681557796096402897798370703746460841949528306,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,405 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 2 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000014,HashKeyRange: {StartingHashKey:
 297747071055821155530452781502797185024,EndingHashKey:
 319014718988379809496913694467282698239},SequenceNumberRange: {StartingSequenceNumber:
 49627894338659257050897872274656834985473812480443547874,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,581 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000008,HashKeyRange: {StartingHashKey:
 170141183460469231731687303715884105728,EndingHashKey:
 191408831393027885698148216680369618943},SequenceNumberRange: {StartingSequenceNumber:
 49627894338525452579706688535807620675837922311407665282,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,586 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 1 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000013,HashKeyRange: {StartingHashKey:
 276479423123262501563991868538311671808,EndingHashKey:
 297747071055821155530452781502797185023},SequenceNumberRange: {StartingSequenceNumber:
 49627894338636956305699341651515299267201164118937567442,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:24,790 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000012,HashKeyRange: {StartingHashKey:
 255211775190703847597530955573826158592,EndingHashKey:
 276479423123262501563991868538311671807},SequenceNumberRange: {StartingSequenceNumber:
 49627894338614655560500811028373763548928515757431587010,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 2
event: 4; timestamp: 1648030809282, 2022-03-23T10:20:09.282Z
event: 3; timestamp: 1648030797697, 2022-03-23T10:19:57.697Z
event: 5; timestamp: 1648030810871, 2022-03-23T10:20:10.871Z
```

```
11:59:24,907 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000011,HashKeyRange: {StartingHashKey:
 233944127258145193631070042609340645376,EndingHashKey:
 255211775190703847597530955573826158591},SequenceNumberRange: {StartingSequenceNumber:
 496278943385923548153022804052322278306558673959256065 78,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 2
event: 7; timestamp: 1648030834105, 2022-03-23T10:20:34.105Z
event: 1; timestamp: 1648030794441, 2022-03-23T10:19:54.441Z
event: 2; timestamp: 1648030796122, 2022-03-23T10:19:56.122Z
event: 8; timestamp: 1648030887171, 2022-03-23T10:21:27.171Z
XXXXXXXXXXXXXX Window with 3 events; Watermark: 1648030809281, 2022-03-23T10:20:09.281Z
3
1
2
XXXXXXXXXXXXXX Window with 2 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
4
5
XXXXXXXXXXXXXX Window with 1 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
6
XXXXXXXXXXXXXX Window with 1 events; Watermark: 1648030887170, 2022-03-23T10:21:27.170Z
7
```

输出只显示 4 个窗口（缺少最后一个包含事件 8 的窗口）。这是由于事件时间和水印策略造成的。最后一个窗口无法关闭，因为使用预构建的水印策略，时间永远不会超过从流中读取的最后一个事件的时间。但是要关闭窗口，时间需要在最后一个事件之后提前 10 秒以上。在这种情况下，最后一个水印是 2022-03-23T10:21:27 .170Z，但为了关闭会话窗口，需要在 10 秒和 1 毫秒后使用水印。

如果`withIdleness`选项从水印策略中移除，任何会话窗口都不会关闭，因为窗口运算符的 “全局水印” 无法前进。

请注意，当 Flink 应用程序启动时（或者存在数据偏差），某些分片的消耗速度可能比其他分片更快。这可能会导致子任务过早发出一些水印（子任务可能会根据一个分片的内容发出水印，而没有从它订阅的其他分片中消耗掉）。缓解的方法是添加安全缓冲区的不同水印策略(`forBoundedOutOfOrderness(Duration.ofSeconds(30))`或者明确允许迟到的事件(`allowedLateness(Time.minutes(5))`.

Apache Flink 状态函数

有状态函数是简化构建分布式有状态应用程序的 API。它基于具有持久状态的函数，可以与强一致性保证进行动态交互。

有状态函数应用程序基本上只是 Apache Flink 应用程序，因此可以部署到 Kinesis Data Analytics。但是，为 Kubernetes 集群和 Kinesis Data Analytics 打包有状态函数之间存在一些区别。有状态函数应用程序最重要的方面是模块配置包含配置状态函数运行时所需的所有运行时信息。此配置通常打包到特定于状态函数的容器中，然后部署在 Kubernetes 上。但是，使用 Kinesis Data Analytics 是不可能的。

以下是对 StateFun Kinesis Data Analytics 的 Python 示例：

Apache Flink 应用程序模板

客户不必将客户容器用于状态函数运行时，而是可以编译一个只调用状态函数运行时并包含所需依赖项的 Flink 应用程序 jar。对于 Flink 1.13，所需的依赖项与以下内容类似：

```
<dependency>
 <groupId>org.apache.flink</groupId>
 <artifactId>statefun-flink-distribution</artifactId>
 <version>3.1.0</version>
 <exclusions>
  <exclusion>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-log4j12</artifactId>
  </exclusion>
  <exclusion>
   <groupId>log4j</groupId>
   <artifactId>log4j</artifactId>
  </exclusion>
 </exclusions>
</dependency>
```

Flink 应用程序调用状态函数运行时的主要方法如下所示：

```
public static void main(String[] args) throws Exception {
 final StreamExecutionEnvironment env =
 StreamExecutionEnvironment.getExecutionEnvironment();

 StatefulFunctionsConfig stateFunConfig = StatefulFunctionsConfig.fromEnvironment(env);

 stateFunConfig.setProvider((StatefulFunctionsUniverseProvider) (classLoader,
 statefulFunctionsConfig) -> {
  Modules modules = Modules.loadFromClassPath();
  return modules.createStatefulFunctionsUniverse(stateFunConfig);
 });

 StatefulFunctionsJob.main(env, stateFunConfig);
}
```

请注意，这些组件是通用的，独立于状态函数中实现的逻辑。

模块配置的位置

有状态函数模块配置需要包含在类路径中才能在有状态函数运行时发现。最好将其包含在 Flink 应用程序的资源文件夹中，然后将其打包到 jar 文件中。

与普通的 Apache Flink 应用程序类似，然后您可以使用 maven 创建一个 uber jar 文件并在 Kinesis Data Analytics 上部署该文件。

适用于 Apache Flink 的 Kinesis Data Analytics 的早期版本信息

> **Note**
>
> 在新区域中，我们仅支持服务可用时及以后的最新 Flink 版本。

本主题包含有关将 Kinesis Data Analytics 与旧 Apache Flink 版本一起使用的信息。Kinesis Data Analytics 所支持的 Apache Flink 版本是1.13.2(建议),1.11.3、1.11.1、1.8.2和1.6.2.

建议您将支持的最新 Apache Flink 版本与 Kinesis Data Analytics 应用程序一起使用。Apache Flink 版本 1.13.2 具有以下功能：

- 对该项的支持Apache Flink 表 API 和 SQL
- Support Python 应用程序。
- Support Java 版本 11 和 Scala 版本 2.12
- 改进的内存模型
- RockSDB 优化以提高应用程序稳定性
- Support Apache Flink 控制面板中的任务管理器和堆栈跟踪。

本主题包含下列部分：

将 Apache Flink Kinesis Streams 连接器与早期 Apache Flink 版本一起使用

Apache Flink Kinesis Streams 连接器在 1.11 版之前没有包含在 Apache Flink 中。要使应用程序将 Apache Flink Kinesis 连接器与早期 Apache Flink 版本一起使用，您必须下载、编译并安装应用程序使用的 Apache Flink 版本。该连接器用于使用来自作为应用程序源的 Kinesis 流的数据，或者将数据写入到用于应用程序输出的 Kinesis 流中。

> **Note**
>
> 确保你正在使用构建连接器KPL 0.14.0或者更高。

要下载并安装 Apache Flink 版本 1.8.2 源代码，请执行以下操作：

1. 确保已安装 Apache Maven，并且 JAVA_HOME 环境变量指向 JDK 而不是 JRE。您可以使用以下命令测试 Apache Maven 安装：

```
mvn -version
```

2. 下载 Apache Flink 版本 1.8.2 源代码：

```
wget https://archive.apache.org/dist/flink/flink-1.8.2/flink-1.8.2-src.tgz
```

3. 解压缩 Apache Flink 源代码：

```
tar -xvf flink-1.8.2-src.tgz
```

4. 转到 Apache Flink 源代码目录：

```
cd flink-1.8.2
```

5. 编译并安装 Apache Flink：

```
mvn clean install -Pinclude-kinesis -DskipTests
```

Note

如果你正在微软 Windows 上编译 Flink，你需要添加-Drat.skip=true参数。

使用 Apache Flink 1.8.2 构建应用程序

本节包含有关用于构建与 Apache Flink 1.8.2 一起使用的 Kinesis Data Analytics 应用程序的组件的信息。

将以下组件版本用于 Kinesis Data Analytics 应用程序：

组件	版本
Java	1.8（建议）
Apache Flink	1.8.2
适用于 Flink 运行时的 Kinesis Data Analytics (aws-kinesisanalytics-runtime)	1.0.1
Kinesis Data Analytics Flink 连接器 (aws-kinesisanalytics-flink)	1.0.1
Apache Maven	3.1

要使用 Apache Flink 1.8.2 编译应用程序，请使用以下参数运行 Maven：

```
mvn package -Dflink.version=1.8.2
```

对于一个例子pom.xml该文件适用于使用 Apache Flink 版本 1.8.2 的 Kinesis Data Analytics 应用程序的适用于 Flink 的 Kinesis Data Analytics 1.8.2 入门应用程序.

有关如何为 Kinesis Data Analytics 应用程序构建和使用应用程序代码的信息，请参阅创建应用程序 (p. 3).

使用 Apache Flink 1.6.2 构建应用程序

本节包含有关用于构建与 Apache Flink 1.6.2 一起使用的 Kinesis Data Analytics 应用程序的组件的信息。

将以下组件版本用于 Kinesis Data Analytics 应用程序：

组件	版本
Java	1.8（建议）
AmazonJava 软件开发工具包	1.11.379
Apache Flink	1.6.2
适用于 Flink 运行时的 Kinesis Data Analytics (aws-kinesisanalytics-runtime)	1.0.1
Kinesis Data Analytics Flink 连接器 (aws-kinesisanalytics-flink)	1.0.1
Apache Maven	3.1
Apache Beam	Apache Flink 1.6.2 不支持。

> **Note**
>
> 使用 Kinesis Data Analytics 运行时版本1.0.1，你可以在你的`pom.xml`文件而不是使用`-Dflink.version`编译应用程序代码时的参数。

对于一个例子`pom.xml`该文件适用于使用 Apache Flink 版本 1.6.2 的 Kinesis Data Analytics 应用程序的适用于 Flink 的 Kinesis Data Analytics 1.6.2 入门应用程序.

有关如何为 Kinesis Data Analytics 应用程序构建和使用应用程序代码的信息，请参阅创建应用程序 (p. 3).

升级应用程序

要升级 Kinesis Data Analytics 应用程序的版本，您必须更新应用程序代码、删除之前的应用程序，然后使用更新的代码创建新应用程序。为此，请执行以下操作：

- 更改 Kinesis Data Analytics 运行时和 Kinesis Data Analytics Flink 连接器的版本（aws-kinesisanalytics-flink) 在你的应用程序`pom.xml`文件到 1.1.0。
- 从应用程序的 `pom.xml` 文件中删除 `flink.version` 属性。在下一步中编译应用程序代码时，您将提供该参数。
- 使用以下命令重新编译应用程序代码：

```
mvn package -Dflink.version=1.13.2
```

- 删除现有的应用程序。再次创建应用程序，然后选择Apache Flink 版本 1.13.2（推荐版本）对于应用程序运行时.

> **Note**
>
> 不能使用以前应用程序版本的快照。

Apache Flink 1.6.2 和 1.8.2 中的可用连接器

Apache Flink 框架包含用于从各种源中访问数据的连接器。

- 有关 Apache Flink 1.6.2 框架中的可用连接器的信息，请参阅连接器 (1.6.2)中的Apache Flink 文档 (1.6.2).
- 有关 Apache Flink 1.8.2 框架中的可用连接器的信息，请参阅连接器 (1.8.2)中的Apache Flink 文档 (1.8.2).

入门：Flink 1.11.3

本主题包含的版本入门 (DataStreamAPI) (p. 76)使用 Apache Flink 1.11.3 的教程。

本节介绍了针对 Apache Flink 的 Kinesis Data Analytics 的基本概念和DataStreamAPI。它介绍了可用于创建和测试应用程序的选项。它还提供了相应的说明以安装所需的工具，以完成本指南中的教程和创建第一个应用程序。

主题

- Flink 应用程序的 Kinesis Data Analytics 的组件 (p. 261)
- 完成练习的先决条件 (p. 261)
- 第 1 步：设置Amazon创建账户和创建管理员用户 (p. 262)
- 第 2 步：设置 Amazon Command Line Interface (Amazon CLI) (p. 264)
- 第 3 步：创建并运行 Apache Flink 应用程序的 Kinesis Data Analytics 应用程序 (p. 265)
- 第 4 步：清理 Amazon 资源 (p. 275)
- 第 5 步：后续步骤 (p. 276)

Flink 应用程序的 Kinesis Data Analytics 的组件

为了处理数据，您的 Kinesis Data Analytics 应用程序使用 Java/Apache Maven 或 Scala 应用程序，该应用程序使用 Apache Flink 运行时处理输入和生成输出。

Kinesis Data Analytics 应用程序具有以下组件：

- 运行时属性：您可以使用运行时属性配置应用程序，而无需重新编译应用程序代码。
- 源：应用程序通过使用数据资源. 源连接器从 Kinesis Data Streams、Amazon S3 存储桶等读取数据。有关更多信息，请参阅 源 (p. 8)。
- 运算符：使用一个或多个应用程序处理数据。运营商. 操作符可以转换、丰富或聚合数据。有关更多信息，请参阅 DataStreamAPI 操作符 (p. 12)。
- 接收器：使用该应用程序将数据生成到外部源。水槽. 接收器连接器将数据写入到 Kinesis 数据流、Kinesis Data Firehose 传输流、Amazon S3 存储桶等。有关更多信息，请参阅 接收器 (p. 9)。

在您创建、编译和打包应用程序代码后，您将代码包上传至 Amazon Simple Storage Service (Amazon S3) 存储桶。然后，您创建 Kinesis Data Analytics 应用程序。您在代码包位置中传入 Kinesis 数据流以作为流数据源，它通常是接收应用程序处理的数据的流或文件位置。

完成练习的先决条件

要完成本指南中的步骤，您必须满足以下条件：

- Java 开发工具包 (JDK) 版本 11. 设置 `JAVA_HOME` 环境变量，使其指向您的 JDK 安装位置。

- 我们建议您使用开发环境（如 Eclipse Java Neon 或 IntelliJ Idea）来开发和编译您的应用程序。
- Git 客户端. 如果尚未安装 Git 客户端，请安装它。
- Apache Maven 编译器插件。Maven 必须位于您的有效路径中。要测试您的 Apache Maven 安装，请输入以下内容：

```
$ mvn -version
```

要开始，请转到第 1 步：设置Amazon创建管理员用户 (p. 77)。

第 1 步：设置Amazon创建账户和创建管理员用户

首次使用 Kinesis Data Analytics 前，请完成以下任务：

1. 注册 Amazon (p. 262)
2. 创建 IAM 用户 (p. 262)

注册 Amazon

当你注册时Amazon，您的账户会自动注册所有 Amazon 服务，包括 Kinesis Data Analytics。您只需为使用的服务付费。

使用 Kinesis Data Analytics 时，您仅需为实际使用的资源付费。如果您是新手Amazon客户，您可以免费试用 Kinesis Data Analytics。有关更多信息，请参阅 Amazon 免费套餐。

如果您已有 Amazon 账户，请跳到下一个任务。如果您没有Amazon账户中，请执行以下步骤创建账户。

创建 Amazon 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。

记下您的账户 ID，因为进行下一个任务时需要用到。

创建 IAM 用户

中的服务Amazon要求您在访问时提供凭证，如 Kinesis Data Analytics（Kinesis Data Analytics）。这样，服务才能确定您是否有权访问该服务所拥有的资源。Amazon Web Services Management Console要求您输入密码。

您可以为您的创建访问密钥Amazon账户以访问Amazon Command Line Interface(Amazon CLI) 或 API。但是，我们不建议使用您的Amazon账户的凭证访问Amazon。相反，我们建议您使用 Amazon Identity and Access Management (IAM)。创建 IAM 用户，将该用户添加到具有管理权限的 IAM 组，然后向您创建的 IAM 用户授予管理权限。您随后便可以使用一个特殊的 URL 和该 IAM 用户的凭证访问 Amazon。

如果你注册了Amazon，但是您尚未为自己创建 IAM 用户，可以使用 IAM 控制台创建一个。

本指南中的入门练习假定您拥有具有管理员权限的用户 (`adminuser`)。请按照以下过程在您的账户中创建 `adminuser`。

为管理员创建组

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Groups (组)，然后选择 Create New Group (创建新组)。
3. 对于组名，输入组的名称，例如 **`Administrators`**，然后选择 下一步。
4. 在策略列表中，选中AdministratorAccess政策。您可以使用 Filter (筛选) 菜单和 Search (搜索) 框来筛选策略列表。
5. 选择 Next Step (下一步)，然后选择 Create Group (创建组)。

您的新组列在 Group Name 下方。

要为您自己创建 IAM 用户，请将用户添加到 Administrator 组，然后创建密码。

1. 在导航窗格中，选择 Users，然后选择 Add user。
2. 在 User name(用户名) 框中，输入一个用户名。
3. 选择这两者以编程方式访问和Amazon管理控制台访问.
4. 选择 Next:。Permissions (下一步：权限)。
5. 选中 Administrators 组旁的复选框。接下来，选择 Next (下一步)：审核。
6. 选择 Create user。

以新 IAM 用户身份登录

1. 注销 Amazon Web Services Management Console。
2. 使用下面的 URL 格式登录控制台：

   `https://aws_account_number.signin.aws.amazon.com/console/`

   这些区域有：*`aws_account_number`*您的账户 ID 是否没有连字符。例如，如果您的账户 ID 是 1234-5678-9012，请替换*`aws_account_number`*和**`123456789012`**. 有关如何查找您的账号的更多信息，请参阅您的Amazon账户 ID 及其别名中的IAM 用户指南.
3. 输入您刚创建的 IAM 用户名和密码。登录后，导航栏将显示 *`your_user_name`* @ *`your_aws_account_id`*。

   Note

   如果您不希望您的登录页面 URL 包含 账户 ID，可以创建账户别名。

创建或删除账户别名

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格上，选择 Dashboard。
3. 查找 IAM 用户登录链接。
4. 要创建别名，请选择 Customize (自定义)。输入要用于别名的名称，然后选择 Yes, Create (是，创建)。
5. 要删除别名，请选择 Customize，然后选择 Yes, Delete。登录 URL 会恢复使用 账户 ID。

要在创建账户别名后登录，请使用以下 URL：

`https://your_account_alias.signin.aws.amazon.com/console/`

要为您的账户验证 IAM 用户的登录链接，请打开 IAM 控制台并在控制面板的 IAM 用户登录链接下进行检查。

有关 IAM 的更多信息，请参阅以下文档：

- Amazon Identity and Access Management (IAM)
- 入门
- IAM 用户指南

下一个步骤

第 2 步：设置 Amazon Command Line Interface (Amazon CLI) (p. 264)

第 2 步：设置 Amazon Command Line Interface (Amazon CLI)

在此步骤中，您将下载并配置Amazon CLI与 Kinesis Data Analytics 一起使用。

Note

本指南中的入门练习假定您使用账户中的管理员凭证 (`adminuser`) 来执行这些操作。

Note

如果您已安装 Amazon CLI，您可能需要升级以获得最新的功能。有关更多信息，请参阅 。安装 Amazon Command Line Interface中的Amazon Command Line Interface用户指南. 要检查 Amazon CLI 的版本，请运行以下命令：

```
aws --version
```

本教程中的练习需要以下 Amazon CLI 版本或更高版本：

```
aws-cli/1.16.63
```

设置 Amazon CLI

1. 下载并配置 Amazon CLI。有关说明，请参阅《Amazon Command Line Interface 用户指南》中的以下主题：

   - 安装 Amazon Command Line Interface
   - 配置 Amazon CLI

2. 在 Amazon CLI `config` 文件中为管理员用户添加一个命名的配置文件。在执行 Amazon CLI 命令时，您将使用此配置文件。有关指定配置文件的更多信息，请参命名配置文件中的Amazon Command Line Interface用户指南.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

   有关可用的列表Amazon地区，请参阅区域和终端节点中的Amazon Web Services 一般参考.

   Note

   本教程中的示例代码和命令使用美国西部（俄勒冈）区域。要使用不同的区域，请将本教程的代码和命令中的区域更改为要使用的区域。

3. 在命令提示符处输入以下帮助命令来验证设置：

```
aws help
```

设置之后Amazon账户和Amazon CLI中，您可以尝试下一个练习中，在其中，您将配置一个示例应用程序并测试end-to-end设置。

下一个步骤

第 3 步：创建并运行 Apache Flink 应用程序的 Kinesis Data Analytics 应用程序 (p. 265)

第 3 步：创建并运行 Apache Flink 应用程序的 Kinesis Data Analytics 应用程序

在本练习中，您将数据流作为源和接收器，创建 Kinesis Data Analytics 应用程序。

本节包含以下步骤：

创建两个 Amazon Kinesis Data Streams

在为本练习创建 Kinesis Data Analytics 应用程序前，请创建两个 Kinesis Data Analytics 应用程序（`ExampleInputStream`和`ExampleOutputStream`)。您的应用程序将这些数据流用于应用程序源和目标流。

您可以使用 Amazon Kinesis 控制台或以下内容创建这些流：Amazon CLI命令。有关控制台说明，请参阅创建和更新数据流中的Amazon Kinesis Data Streams 开发人员指南.

创建数据流 (Amazon CLI)

1. 创建第一个直播 (`ExampleInputStream`)，请使用以下 Amazon Kinesis`create-stream` Amazon CLI命令。

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. 要创建应用程序用来写入输出的第二个流，请运行同一命令（将流名称更改为 `ExampleOutputStream`）。

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入流，以供应用程序处理。

> **Note**
>
> 此部分需要 Amazon SDK for Python (Boto)。

1. 使用以下内容创建名为 `stock.py` 的文件：

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

2. 在本教程的后面部分，您运行 `stock.py` 脚本，以将数据发送到应用程序。

```
$ python stock.py
```

下载并检查 Apache Flink 流式处理 Java 代码

此示例的 Java 应用程序代码可从以下网址获得。GitHub. 要下载应用程序代码，请执行以下操作：

1. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

2. 导航到 `amazon-kinesis-data-analytics-java-examples/GettingStarted`目录。

请注意有关应用程序代码的以下信息：

- 一个项目对象模型 (pom.xml)此文件包含有关应用程序的配置和依赖关系（包括 Kinesis Data Analytics 库）的信息。
- `BasicStreamingJob.java` 文件包含定义应用程序功能的 `main` 方法。

- 该应用程序使用 Kinesis 源从源流中进行读取。以下代码段创建 Kinesis 源：

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- 您的应用程序使用 `StreamExecutionEnvironment` 对象创建源和接收连接器以访问外部资源。
- 该应用程序将使用静态属性创建源和接收连接器。要使用动态应用程序属性，请使用 `createSourceFromApplicationProperties` 和 `createSinkFromApplicationProperties` 方法以创建连接器。这些方法读取应用程序的属性来配置连接器。

  有关运行时属性的更多信息，请参阅运行时属性 (p. 18)。

编译应用程序代码

在本节中，您使用 Apache Maven 编译器创建应用程序的 Java 代码。有关安装 Apache Maven 和 Java 开发工具包 (JDK) 的信息，请参阅完成练习的先决条件 (p. 76)。

编译应用程序代码

1. 要使用您的应用程序代码，您将其编译和打包成 JAR 文件。您可以通过两种方式之一编译和打包您的代码：

   - 使用命令行 Maven 工具。在包含 `pom.xml` 文件的目录中通过运行以下命令创建您的 JAR 文件：

     ```
     mvn package -Dflink.version=1.11.3
     ```

   - 设置开发环境。有关详细信息，请参阅您的开发环境文档。

     Note

     提供的源代码依赖于 Java 11 中的库。确保项目的 Java 版本为 11。

   您可以作为 JAR 文件上传您的包，也可以将包压缩为 ZIP 文件并上传。如果您使用 Amazon CLI 创建应用程序，您可以指定您的代码内容类型（JAR 或 ZIP）。
2. 如果编译时出错，请验证 `JAVA_HOME` 环境变量设置正确。

如果应用程序成功编译，则创建以下文件：

`target/aws-kinesis-analytics-java-apps-1.0.jar`

上传 Apache Flink 流式处理 Java 代码

在本节中，您创建 Amazon Simple Storage Service (Amazon S3) 存储桶并上传应用程序代码。

上传应用程序代码

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. Enter**`ka-app-code-<username>`**中的Bucket name字段中返回的子位置类型。将后缀（如您的用户名）添加到存储桶名称，以使其具有全局唯一性。请选择 Next (下一步)。
4. 在配置选项步骤中，让设置保持原样，然后选择下一步。
5. 在设置权限步骤中，让设置保持原样，然后选择下一步。
6. 请选择 Create bucket (创建存储桶)。
7. 在 Amazon S3 控制台中，选择ka-app-code-***`<username>`***存储桶，然后选择上传.

8. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `aws-kinesis-analytics-java-apps-1.0.jar` 文件。请选择 Next (下一步)。
9. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

现在，您的应用程序代码存储在 Amazon S3 存储桶中，应用程序可以在其中访问代码。

创建并运行 Kinesis Data Analytics 应用程序

您可以使用控制台或创建并运行 Kinesis Data Analytics 应用程序。Amazon CLI.

> **Note**
>
> 当您使用控制台创建应用程序时，您的Amazon Identity and Access Management(IAM) 和亚马逊CloudWatch日志资源是为你创建的。当您使用 Amazon CLI 创建应用程序时，您可以单独创建这些资源。

主题

创建并运行应用程序（控制台）

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于描述，输入 **`My java test app`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。
   - 将版本下拉列表保留为Aache Flink 1.11（推荐版本）.
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
5. 选择 Create application（创建应用程序）。

> **Note**
>
> 使用控制台创建 Kinesis Data Analytics 应用程序时，您可以选择为您的应用程序创建 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源使用您的应用程序名称和区域命名，如下所示：
>
> - 策略：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - 角色：`kinesis-analytics-`*`MyApplication`*`-`*`us-west-2`*

编辑 IAM 策略

编辑 IAM 策略以添加访问 Kinesis Data Streams 的权限。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。

2. 选择策略。选择控制台在上一部分中为您创建的 **kinesis-analytics-service-MyApplication-us-west-2** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。
4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
```

```
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

配置应用程序

1. 在存储库的MyApplication在页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**ka-app-code-*<username>***.
   - 适用于Amazon S3 对象的路径输入**aws-kinesis-analytics-java-apps-1.0.jar**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
4. 在 Properties (属性) 下，对于 Group ID (组 ID)，输入 **ProducerConfigProperties**。
5. 输入以下应用程序属性和值：

密钥	值
flink.inputstream.initpos	**LATEST**
aws.region	**us-west-2**
AggregationEnabled	**false**
group.id	**ProducerConfigProperties**

6. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
7. 适用于CloudWatch记录在中，选择启用”复选框。
8. 选择 Update (更新) 。

Note

当您选择启用亚马逊时CloudWatch日志记录中，Kinesis Data Analytics 会为您创建日志组和日志流。这些资源的名称如下所示：

- 日志组：`/aws/kinesis-analytics/MyApplication`
- 日志流：`kinesis-analytics-log-stream`

运行应用程序

可以通过运行应用程序、打开 Apache Flink 控制面板并选择所需的 Flink 作业来查看 Flink 作业图。

停止应用程序

在存储库的MyApplication在页面上，选择停止. 确认该操作。

更新应用程序

使用控制台，您可以更新应用程序设置，例如应用程序属性、监控设置，或应用程序 JAR 文件的位置和文件名。如果您需要更新应用程序代码，还可以从 Amazon S3 存储桶重新加载应用程序 JAR。

在存储库的MyApplication在页面上，选择配置. 更新应用程序设置，然后选择更新。

创建并运行应用程序 (Amazon CLI)

在本部分中，您将使用Amazon CLI创建和运行 Kinesis Data Analytics 应用程序。Apache Flink 的 Kinesis Data Analytics 使用`kinesisanalyticsv2` Amazon CLI命令来创建 Kinesis Data Analytics 应用程序并与之交互。

创建权限策略

> **Note**
>
> 您必须为应用程序创建一个权限策略和角色。如果未创建这些 IAM 资源，应用程序将无法访问其数据和日志流。

首先，使用两个语句创建权限策略：一个语句授予对源流执行 `read` 操作的权限，另一个语句授予对接收器流执行 `write` 操作的权限。然后，您将策略附加到 IAM 角色（在下一部分中创建）。因此，在 Kinesis Data Analytics 代入该角色时，服务具有必要的权限从源流进行读取和写入接收器流。

使用以下代码创建 `KAReadSourceStreamWriteSinkStream` 权限策略。Replace*username*使用您用于创建 Amazon S3 存储桶以存储应用程序代码的用户名。将 Amazon 资源名称 (ARN) 中的账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleOutputStream"
        }
    ]
}
```

适用于step-by-step有关创建权限策略的说明，请参阅教程：创建并附加您的第一个客户托管策略中的IAM 用户指南.

> **Note**
>
> 要访问其他亚马逊服务，您可以使用Amazon SDK for Java. Kinesis Data Analytics 会自动将开发工具包所需的凭证设置为与您的应用程序关联的服务执行 IAM 角色的凭证。无需执行其他步骤。

创建 IAM 角色

在本节中，您创建一个 IAM 角色，Kinesis Data Analytics 应用程序可以代入此角色来读取源流和写入接收器流。

没有权限，Kinesis Data Analytics 无法访问您的流。您通过 IAM 角色授予这些权限。每个 IAM 角色附加了两种策略。此信任策略授予 Kinesis Data Analytics 代入该角色的权限，权限策略确定 Kinesis Data Analytics 入该角色后可以执行的操作。

您将在上一部分中创建的权限策略附加到此角色。

创建 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles (角色) 和 Create Role (创建角色)。
3. 在选择受信任实体的类型，选择Amazon服务. 在 Choose the service that will use this role (选择将使用此角色的服务) 下，选择 Kinesis。在选择您的使用案例下，选择 Kinesis Analytics。

   选择 Next:。Permissions (下一步：权限)。
4. 在存储库的附加权限策略在页面上，选择后续：审核。在创建角色后，您可以附加权限策略。
5. 在存储库的创建角色页面，输入**`KA-stream-rw-role`**(对于)Role name (角色名称). 请选择 Create role（创建角色）。

   现在，您已经创建了一个名为的新 IAM 角色`KA-stream-rw-role`. 接下来，您更新角色的信任和权限策略。
6. 将权限策略附加到角色。

   > **Note**
   >
   > 在本练习中，Kinesis Data Analytics 代入此角色，以便同时从 Kinesis 数据流（源）读取数据和将输出写入另一个 Kinesis 数据流。因此，您附加在上一步（the section called “创建权限策略” (p. 271)）中创建的策略。

   a. 在 Summary (摘要) 页上，选择 Permissions (权限) 选项卡。
   b. 选择附加策略。
   c. 在搜索框中，输入 **`KAReadSourceStreamWriteSinkStream`**（您在上一部分中创建的策略）。
   d. 选择KAReadSourceStreamWriteSinkStream策略，然后选择附加策略.

现在，您已经创建了应用程序用来访问资源的服务执行角色。记下新角色的 ARN。

适用于step-by-step有关创建角色的说明，请参阅创建 IAM 角色 (控制台)中的IAM 用户指南.

创建 Kinesis Data Analytics 应用程序

1. 将以下 JSON 代码保存到名为 `create_request.json` 的文件中。将示例角色 ARN 替换为您之前创建的角色的 ARN。将存储桶 ARN 后缀 (*`username`*) 替换为在前一部分中选择的后缀。将服务执行角色中的示例账户 ID (*`012345678901`*) 替换为您的账户 ID。

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_11",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/KA-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
```

```
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. 使用上述请求执行 `CreateApplication` 操作来创建应用程序：

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

应用程序现已创建。您在下一步中启动应用程序。

启动应用程序

在本节中，您使用 `StartApplication` 操作来启动应用程序。

启动应用程序

1. 将以下 JSON 代码保存到名为 `start_request.json` 的文件中。

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. 使用上述请求执行 `StartApplication` 操作来启动应用程序：

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

应用程序正在运行。您可以在亚马逊上查看 Kinesis Data Analytics 指标CloudWatch控制台以验证应用程序是否正常工作。

停止应用程序

在本节中，您使用 `StopApplication` 操作来停止应用程序。

停止应用程序

1. 将以下 JSON 代码保存到名为 `stop_request.json` 的文件中。

```
{
    "ApplicationName": "test"
}
```

2. 使用下面的请求执行 `StopApplication` 操作来停止应用程序：

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

应用程序现已停止。

添加 CloudWatch 日志记录选项

您可以使用Amazon CLI添加亚马逊CloudWatch将流记录到应用程序中。有关使用的信息CloudWatch有应用程序的日志，请参阅the section called “设置日志记录” (p. 205).

更新环境属性

在本节中，您使用 `UpdateApplication` 操作更改应用程序的环境属性，而无需重新编译应用程序代码。在本示例中，您更改源流和目标流的区域。

更新应用程序的环境属性

1. 将以下 JSON 代码保存到名为 `update_properties_request.json` 的文件中。

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. 使用前面的请求执行 `UpdateApplication` 操作以更新环境属性：

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

更新应用程序代码

在您需要使用新版本的代码包更新应用程序代码时，您可以使用 `UpdateApplication` Amazon CLI 操作。

Note

要使用相同的文件名加载新版本的应用程序代码，您必须指定新的对象版本。有关使用 Amazon S3 对象版本的更多信息，请参阅启用或禁用版本控制.

使用Amazon CLI，请从 Amazon S3 存储桶中删除以前的代码包，上传新版本，然后调用并调用`UpdateApplication`指定相同的 Amazon S3 存储桶和对象名称以及新的对象版本。应用程序将使用新的代码包重新启动。

以下示例 `UpdateApplication` 操作请求重新加载应用程序代码并重新启动应用程序。将 `CurrentApplicationVersionId` 更新为当前的应用程序版本。您可以使用 `ListApplications` 或 `DescribeApplication` 操作检查当前的应用程序版本。将存储桶名称后缀 (*<username>*) 更新为在the section called “创建两个 Amazon Kinesis Data Streams” (p. 265)一节中选择的后缀。

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

下一个步骤

第 4 步：清理 Amazon 资源 (p. 275)

第 4 步：清理 Amazon 资源

本节包含清理在入门教程中创建的 Amazon 资源的过程。

本主题包含下列部分：

- 删除 Kinesis Data Analytics 应用程序 (p. 275)
- 删除 Kinesis Data Streams (p. 275)
- 删除您的 Amazon S3 对象和存储桶 (p. 276)
- 删除 IAM 资源 (p. 276)
- 删除 CloudWatch 资源 (p. 276)
- 下一个步骤 (p. 276)

删除 Kinesis Data Analytics 应用程序

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 在应用程序的页面中，选择 Delete (删除)，然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics.

2. 在 Kinesis Data Streams 面板中，选择ExampleInput流.
3. 在ExampleInput流在页面上，选择删除 Kinesis Streams然后确认删除。
4. 在Kinesis Streams在页面上，选择ExampleOutput流，选择操作，选择Delete，然后确认删除。

删除您的 Amazon S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择ka-app-code-***<username>***存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除 IAM 资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-***<your-region>***政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏中，选择 Roles（角色）。
7. 选择Kinesis-AnalyticsMyApplication-***<your-region>***角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除 CloudWatch 资源

1. 打开 CloudWatch 控制台 (https://console.aws.amazon.com/cloudwatch/)。
2. 在导航栏中，选择 Logs (日志)。
3. 选择/aws/kinesis-析ics/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

下一个步骤

第 5 步：后续步骤 (p. 276)

第 5 步：后续步骤

现在您已创建并运行基本的 Kinesis Data Analytics 应用程序，请参阅以下资源了解更高级的 Kinesis Data Analytics 解决方案。

- 这些区域有：AmazonAmazon Kinesis 的流数据解决方案：这些区域有：Amazon适用于 Amazon Kinesis 的流数据解决方案自动配置Amazon轻松捕获、存储、处理和交付流数据所需的服务。该解决方案为解决流式数据使用案例提供了多种选择。Kinesis Data Analytics 选项提供了end-to-end直播 ETL 示例演示了对模拟的纽约出租车数据运行分析操作的真实世界应用程序。该解决方案设置了所有必要Amazon资源，例如 IAM 角色和策略，CloudWatch以及控制面板CloudWatch警报。
- AmazonAmazon MSK 流式传输数据解决方案：这些区域有：Amazon亚马逊 MSK 流数据解决方案提供 Amazon CloudFormation数据流经生产者、流式存储、消费者和目的地的模板。
- 使用 Apache Flink 和 Apache Kafka 的点击流实验室：一个端到端的实验室，该实验室用于点击流使用案例，使用适用于 Apache Kafka 的亚马逊托管流媒体进行流媒体存储和适用于 Apache Flink 的 Amazon Kinesis Data Analytics 应用程序用于流

- 直播分析研讨会：在本研讨会中，你建立了end-to-end用于近乎实时地采集、分析和可视化流数据的流式体系结构。你开始改善纽约市一家出租车公司的运营。您可以近乎实时地分析纽约市出租车车队的遥测数据，以优化其车队运营。
- Kinesis Data Analytics 示例 (p. 123)：此开发人员指南中提供在 Kinesis Data Analytics 中创建和使用应用程序的示例。它们包括示例代码和step-by-step说明，以帮助您创建 Kinesis Data Analytics 应用程序和测试结果。
- 学习 Flink: 入门训练：官方入门 Apache Flink 培训，帮助您开始编写可扩展的流式 ETL、分析和事件驱动型应用程序。

  Note

  请注意，Kinesis Data Analytics 不支持本培训中使用的 Apache Flink 版本 (1.12)。你可以在 Flink Kinesis Data Analytics 中使用 Flink 1.13。
- Apache Flink 代码示例：一个GitHub存储了各种 Apache Flink 应用程序示例的存储库。

入门：Flink 1.8.2

本主题包含入门 (DataStreamAPI) (p. 76)使用 Apache Flink 1.8.2 的教程。

主题

Flink 应用程序的 Kinesis Data Analytics 的组件

为了处理数据，您的 Kinesis Data Analytics 应用程序使用 Java/Apache Maven 或 Scala 应用程序，该应用程序使用 Apache Flink 运行时处理输入和生成输出。

Kinesis Data Analytics 应用程序具有以下组件：

- 运行时属性：您可以使用运行时属性配置应用程序，而无需重新编译应用程序代码。
- 源：应用程序通过使用资源. 源连接器从 Kinesis Data Streams、Amazon S3 存储桶等读取数据。有关更多信息，请参阅 源 (p. 8)。
- 运算符：应用程序使用一个或多个方法处理数据。运营商. 操作符可以转换、丰富或聚合数据。有关更多信息，请参阅 DataStreamAPI 操作符 (p. 12)。
- 接收器：此应用程序使用以下方法将数据生成到外水槽. 接收器连接器将数据写入 Kinesis Data Streams、Kinesis Data Firehose 传输流、Amazon S3 存储桶等。有关更多信息，请参阅 接收器 (p. 9)。

在创建、编译和打包应用程序代码后，您将代码包上传到 Amazon Simple Storage Service (Amazon S3) 存储桶。然后，您创建 Kinesis Data Analytics 应用程序。您在代码包位置中传入 Kinesis 数据流以作为流数据源，它通常是接收应用程序处理的数据的流或文件位置。

完成练习的先决条件

要完成本指南中的步骤，您必须满足以下条件：

- Java 开发工具包 (JDK) 版本 8。设置 `JAVA_HOME` 环境变量，使其指向您的 JDK 安装位置。

- 要在本教程中使用 Apache Flink Kinesis 连接器，您必须下载并安装 Apache Flink。有关详细信息，请参阅 将 Apache Flink Kinesis Streams 连接器与早期 Apache Flink 版本一起使用 (p. 258)。
- 我们建议您使用开发环境（如 Eclipse Java Neon 或 IntelliJ Idea）来开发和编译您的应用程序。
- Git 客户端. 如果尚未安装 Git 客户端，请安装它。
- Apache Maven 编译器插件。Maven 必须位于您的有效路径中。要测试您的 Apache Maven 安装，请输入以下内容：

```
$ mvn -version
```

要开始，请转到第 1 步：设置Amazon创建账户并创建管理员用户 (p. 278)。

第 1 步：设置Amazon创建账户并创建管理员用户

首次使用 Kinesis Data Analytics 前，请完成以下任务：

1. 注册 Amazon (p. 278)
2. 创建 IAM 用户 (p. 278)

注册 Amazon

当你注册时Amazon，您的Amazon在中为账户自动注册所有服务Amazon包括 Kinesis Data Analytics。您只需为使用的服务付费。

使用 Kinesis Data Analytics 时，您仅需为实际使用的资源付费。如果您是新手Amazon客户，您可以免费试用 Kinesis Data Analytics。有关更多信息，请参阅 Amazon 免费套餐。

如果您已有 Amazon 账户，请跳到下一个任务。如果您还没有账户，请执行以下步骤创建。

创建 Amazon 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。

注意您的Amazon账户 ID 是因为进行下一个任务时需要用到。

创建 IAM 用户

在中的服务Amazon（如 Kinesis Data Analytics）要求您在访问时提供凭证。这样，服务才能确定您是否有权访问该服务所拥有的资源。Amazon Web Services Management Console要求您输入密码。

您可以为您的创建访问密钥Amazon账户以访问Amazon Command Line Interface(Amazon CLI) 或者 API。但是，我们不建议使用您的Amazon账户的凭证访问Amazon。相反，我们建议您使用 Amazon Identity and Access Management (IAM)。创建 IAM 用户，将该用户添加到具有管理权限的 IAM 组，然后向您创建的 IAM 用户授予管理权限。您随后便可以使用一个特殊的 URL 和该 IAM 用户的凭证访问 Amazon。

如果你注册了Amazon，但是您尚未为自己创建 IAM 用户，您可以使用 IAM 控制台自行创建。

本指南中的入门练习假定您拥有具有管理员权限的用户 (`adminuser`)。请按照以下过程在您的账户中创建 `adminuser`。

为管理员创建组

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Groups (组)，然后选择 Create New Group (创建新组)。
3. 对于组名，输入组的名称，例如 **`Administrators`**，然后选择 下一步。
4. 在策略列表中，选中AdministratorAccess政策。您可以使用 Filter (筛选) 菜单和 Search (搜索) 框来筛选策略列表。
5. 选择 Next Step (下一步)，然后选择 Create Group (创建组)。

您的新组列在 Group Name 下方。

要为您自己创建 IAM 用户，请将其添加到 Administrator 组中，然后创建密码

1. 在导航窗格中，选择 Users，然后选择 Add user。
2. 在 User name(用户名) 框中，输入一个用户名。
3. 选择这两者以编程方式访问和Amazon访问管理控制台.
4. 选择 Next:。Permissions (下一步：权限)。
5. 选中 Administrators 组旁的复选框。接下来，选择 Next (下一步)：审核。
6. 选择 Create user。

以新 IAM 用户身份登录

1. 注销 Amazon Web Services Management Console。
2. 使用下面的 URL 格式登录控制台：

   `https://aws_account_number.signin.aws.amazon.com/console/`

   这些区域有：*`aws_account_number`*您的账户 ID 是否没有连字符。例如，如果您的账户 ID 是 1234-5678-9012，请替换*`aws_account_number`*和**`123456789012`**. 有关如何查找您的账号的更多信息，请参阅您的Amazon账户 ID 及其别名中的IAM 用户指南.
3. 输入您刚创建的 IAM 用户名和密码。登录后，导航栏将显示 *`your_user_name`* @ *`your_aws_account_id`*。

   Note

   如果您不希望您的登录页面 URL 包含 账户 ID，可以创建账户别名。

创建或删除账户别名

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格上，选择 Dashboard。
3. 查找 IAM 用户登录链接。
4. 要创建别名，请选择 Customize (自定义)。输入要用于别名的名称，然后选择 Yes, Create (是，创建)。
5. 要删除别名，请选择 Customize，然后选择 Yes, Delete。登录 URL 会恢复使用 Amazon 账户 ID。

要在创建账户别名后登录，请使用以下 URL：

`https://your_account_alias.signin.aws.amazon.com/console/`

要为您的账户验证 IAM 用户的登录链接，请打开 IAM 控制台并在控制面板的 IAM 用户登录链接下进行检查。

有关 IAM 的更多信息，请参阅以下文档：

- Amazon Identity and Access Management (IAM)
- 入门
- IAM 用户指南

下一个步骤

第 2 步：设置 Amazon Command Line Interface (Amazon CLI) (p. 280)

第 2 步：设置 Amazon Command Line Interface (Amazon CLI)

在此步骤中，您将下载并配置Amazon CLI与 Kinesis Data Analytics 一起使用。

Note

本指南中的入门练习假定您使用账户中的管理员凭证 (`adminuser`) 来执行这些操作。

Note

如果您已安装 Amazon CLI，您可能需要升级以获得最新的功能。有关更多信息，请参阅 。安装 Amazon Command Line Interface中的Amazon Command Line Interface用户指南. 要检查 Amazon CLI 的版本，请运行以下命令：

```
aws --version
```

本教程中的练习需要以下 Amazon CLI 版本或更高版本：

```
aws-cli/1.16.63
```

设置 Amazon CLI

1. 下载并配置 Amazon CLI。有关说明，请参阅《Amazon Command Line Interface 用户指南》中的以下主题：
   - 安装 Amazon Command Line Interface
   - 配置 Amazon CLI
2. 在 Amazon CLI `config` 文件中为管理员用户添加一个命名的配置文件。在执行 Amazon CLI 命令时，您将使用此配置文件。有关指定配置文件的更多信息，请参命名配置文件中的Amazon Command Line Interface用户指南.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

   有关可用区域的列表，请参阅区域和终端节点中的Amazon Web Services 一般参考.

   Note

   本教程中的示例代码和命令使用美国西部（俄勒冈）区域。要使用不同的Amazon将本教程的代码和命令中的区域更改为要使用的区域。

3. 在命令提示符处输入以下帮助命令来验证设置：

```
aws help
```

在你设置之后Amazon账户和Amazon CLI，您可以尝试下一个练习中，在其中，您将配置一个示例应用程序并测试end-to-end设置。

下一个步骤

第 3 步：为 Apache Flink 应用程序创建并运行 Kinesis Data Analytics 应用程序 (p. 281)

第 3 步：为 Apache Flink 应用程序创建并运行 Kinesis Data Analytics 应用程序

在本练习中，您创建一个 Kinesis Data Analytics 应用程序，它将数据流作为源和接收器。

本节包含以下步骤：

创建两个 Amazon Kinesis Data Streams

在为本练习创建 Kinesis Data Analytics 应用程序前，请创建两个 Kinesis Data Analytics 应用程序（`ExampleInputStream`和`ExampleOutputStream`)。您的应用程序将这些数据流用于应用程序源和目标流。

您可以使用 Amazon Kinesis Console 或以下命令创建这些流：Amazon CLI命令。有关控制台说明，请参阅创建和更新数据流中的Amazon Kinesis Data Streams 开发人员指南.

创建数据流 (Amazon CLI)

1. 要创建第一个流 (`ExampleInputStream`)，请使用以下 Amazon Kinesis`create-stream` Amazon CLI命令。

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. 要创建应用程序用来写入输出的第二个流，请运行同一命令（将流名称更改为 `ExampleOutputStream`）。

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
```

```
--region us-west-2 \
--profile adminuser
```

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入流，以供应用程序处理。

Note

此部分需要 Amazon SDK for Python (Boto)。

1. 使用以下内容创建名为 `stock.py` 的文件：

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

2. 在本教程的后面部分，您运行 `stock.py` 脚本，以将数据发送到应用程序。

```
$ python stock.py
```

下载并检查 Apache Flink 流式处理 Java 代码

此示例的 Java 应用程序代码可从GitHub. 要下载应用程序代码，请执行以下操作：

1. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

2. 导航到 `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_8`目录。

请注意有关应用程序代码的以下信息：

- 一个项目对象模型 (pom.xml)文件包含有关应用程序的配置和依赖关系（包括 Kinesis Data Analytics 库）的信息。
- `BasicStreamingJob.java` 文件包含定义应用程序功能的 `main` 方法。
- 该应用程序使用 Kinesis Source 从源流中进行读取。以下代码段创建 Kinesis 源：

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- 您的应用程序使用 `StreamExecutionEnvironment` 对象创建源和接收连接器以访问外部资源。
- 该应用程序将使用静态属性创建源和接收连接器。要使用动态应用程序属性，请使用 `createSourceFromApplicationProperties` 和 `createSinkFromApplicationProperties` 方法以创建连接器。这些方法读取应用程序的属性来配置连接器。

  有关运行时属性的更多信息，请参阅运行时属性 (p. 18)。

编译应用程序代码

在本节中，您使用 Apache Maven 编译器创建应用程序的 Java 代码。有关安装 Apache Maven 和 Java 开发工具包 (JDK) 的信息，请参阅完成练习的先决条件 (p. 277)。

> Note
>
> 要将 Kinesis Analytics 连接器与 1.11 以前的版本结合使用，您需要下载、构建和安装 Apache Maven。有关更多信息，请参阅the section called “将 Apache Flink Kinesis Streams 连接器与早期 Apache Flink 版本一起使用” (p. 258)。

编译应用程序代码

1. 要使用您的应用程序代码，您将其编译和打包成 JAR 文件。您可以通过两种方式之一编译和打包您的代码：

   - 使用命令行 Maven 工具。在包含 `pom.xml` 文件的目录中通过运行以下命令创建您的 JAR 文件：

     ```
     mvn package -Dflink.version=1.8.2
     ```

   - 设置开发环境。有关详细信息，请参阅您的开发环境文档。

     > Note
     >
     > 提供的源代码依赖于 Java 1.8 中的库。确保项目的 Java 版本为 1.8。

   您可以作为 JAR 文件上传您的包，也可以将包压缩为 ZIP 文件并上传。如果您使用 Amazon CLI 创建应用程序，您可以指定您的代码内容类型（JAR 或 ZIP）。
2. 如果编译时出错，请验证 `JAVA_HOME` 环境变量设置正确。

如果应用程序成功编译，则创建以下文件：

`target/aws-kinesis-analytics-java-apps-1.0.jar`

上传 Apache Flink 流式处理 Java 代码

在本节中，您创建 Amazon Simple Storage Service (Amazon S3) 存储桶并上传应用程序代码。

上传应用程序代码

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. Enter**ka-app-code-*<username>***中的Bucket name字段中返回的子位置类型。将后缀（如您的用户名）添加到存储桶名称，以使其具有全局唯一性。请选择 Next (下一步)。
4. 在配置选项步骤中，让设置保持原样，然后选择下一步。
5. 在设置权限步骤中，让设置保持原样，然后选择下一步。
6. 请选择 Create bucket (创建存储桶)。
7. 在 Amazon S3 控制台中，选择ka-app-code-***<username>***存储桶，然后选择上传.
8. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `aws-kinesis-analytics-java-apps-1.0.jar` 文件。请选择 Next (下一步)。
9. 您无需更改该对象的任何设置，因此，请选择 Upload (上传)。

您的应用程序代码现在存储在 Amazon S3 存储桶中，应用程序可以在其中访问代码。

创建并运行 Kinesis Data Analytics 应用程序

您可以使用控制台或创建并运行 Kinesis Data Analytics 应用程序。Amazon CLI.

> Note
>
> 使用控制台创建应用程序时，您的Amazon Identity and Access Management(IAM) 和亚马逊CloudWatch日志资源是为你创建的。当您使用 Amazon CLI 创建应用程序时，您可以单独创建这些资源。

主题

创建并运行应用程序（控制台）

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Analytics 仪表板上，选择创建分析应用.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于描述，输入 **`My java test app`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。
   - 将版本下拉列表保留为 Apache Flink 1.8 (Recommended Version) (Apache Flink 1.8 (建议的版本))。
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
5. 选择 Create application（创建应用程序）。

> Note
>
> 在使用控制台创建 Kinesis Data Analytics 应用程序时，您可以选择为应用程序创建 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源使用您的应用程序名称和区域命名，如下所示：

- 策略：kinesis-analytics-service-*MyApplication*-*us-west-2*
- 角色：kinesis-analytics-*MyApplication*-*us-west-2*

编辑 IAM 策略

编辑 IAM 策略以添加访问 Kinesis Data Streams 的权限。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择策略。选择控制台在上一部分中为您创建的 **kinesis-analytics-service-MyApplication-us-west-2** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。
4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
```

```
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**ka-app-code-*<username>***.
   - 适用于指向 Amazon S3 对象的路径输入**aws-kinesis-analytics-java-apps-1.0.jar**.
3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
4. 在 Properties (属性) 下，对于 Group ID (组 ID)，输入 **ProducerConfigProperties**。
5. 输入以下应用程序属性和值：

密钥	值
flink.inputstream.initpos	**LATEST**
aws.region	**us-west-2**
AggregationEnabled	**false**

6. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
7. 适用于CloudWatch记录在中，选择启用”复选框。
8. 选择 Update (更新) 。

> **Note**
>
> 当您选择启用亚马逊时CloudWatch记录时，Kinesis Data Analytics 会为您创建日志组和日志流。这些资源的名称如下所示：
>
> - 日志组：`/aws/kinesis-analytics/MyApplication`
> - 日志流：`kinesis-analytics-log-stream`

运行应用程序

1. 在存储库的MyApplication页面上，选择运行. 确认该操作。
2. 当应用程序正在运行时，请刷新页面。控制台将显示 Application graph (应用程序图表)。

停止应用程序

在存储库的MyApplication页面上，选择停止. 确认该操作。

更新应用程序

使用控制台，您可以更新应用程序设置，例如应用程序属性、监控设置，或应用程序 JAR 文件的位置和文件名。如果您需要更新应用程序代码，还可以从 Amazon S3 存储桶重新加载应用程序 JAR。

在存储库的MyApplication页面上，选择配置. 更新应用程序设置，然后选择更新。

创建并运行应用程序 (Amazon CLI)

在本部分中，您将使用Amazon CLI创建并运行 Kinesis Data Analytics 应用程序。Apache Flink 的 Kinesis Data Analytics 使用`kinesisanalyticsv2` Amazon CLI命令来创建 Kinesis Data Analytics 应用程序并与之交互。

创建权限策略

> **Note**
>
> 您必须为应用程序创建一个权限策略和角色。如果未创建这些 IAM 资源，应用程序将无法访问其数据和日志流。

首先，使用两个语句创建权限策略：一个语句授予对源流执行 `read` 操作的权限，另一个语句授予对接收器流执行 `write` 操作的权限。然后，您将策略附加到 IAM 角色（在下一部分中将创建此角色）。因此，在 Kinesis Data Analytics 代入该角色时，服务具有必要的权限从源流进行读取和写入接收器流。

使用以下代码创建 `KAReadSourceStreamWriteSinkStream` 权限策略。Replace*username*使用您用于创建 Amazon S3 存储桶以存储应用程序代码的用户名。将 Amazon 资源名称 (ARN) 中的账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleOutputStream"
        }
    ]
}
```

适用于step-by-step有关创建权限策略的说明，请参阅教程：创建并附加您的第一个客户托管策略中的IAM 用户指南.

Note

要访问其他亚马逊服务，您可以使用Amazon SDK for Java. Kinesis Data Analytics 会自动将开发工具包所需的凭证设置为与您的应用程序关联的服务执行 IAM 角色的凭证。无需执行其他步骤。

创建 IAM 角色

在本节中，您将创建一个 IAM 角色，Kinesis Data Analytics 应用程序可以代入此角色，以读取源流并写入接收器流。

如果没有权限，Kinesis Data Analytics 无法访问您的流 您通过 IAM 角色授予这些权限。每个 IAM 角色附加了两个策略。此信任策略授予 Kinesis Data Analytics 该角色的权限，权限策略确定代入该角色后可以执行的操作。

您将在上一部分中创建的权限策略附加到此角色。

创建 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles (角色) 和 Create Role (创建角色)。
3. UNDER选择受信任实体的类型，选择Amazon服务. 在 Choose the service that will use this role (选择将使用此角色的服务) 下，选择 Kinesis。在选择您的使用案例下，选择 Kinesis Analytics。

   选择 Next:。Permissions (下一步：权限)。
4. 在存储库的附加权限策略页面上，选择后续：审核。在创建角色后，您可以附加权限策略。
5. 在存储库的创建角色页面，输入**`KA-stream-rw-role`**(对于)Role name (角色名称). 请选择 Create role（创建角色）。

   现在，您已经创建了一个名为的新 IAM 角色`KA-stream-rw-role`. 接下来，您更新角色的信任和权限策略。
6. 将权限策略附加到角色。

   Note

   对于本练习，Kinesis Data Analytics 代入此角色，以便同时从 Kinesis 数据流（源）读取数据和将输出写入另一个 Kinesis 数据流。因此，您附加在上一步（ the section called “创建权限策略” (p. 287) ）中创建的策略。

   a. 在 Summary (摘要) 页上，选择 Permissions (权限) 选项卡。
   b. 选择附加策略。
   c. 在搜索框中，输入 **`KAReadSourceStreamWriteSinkStream`**（您在上一部分中创建的策略）。
   d. 选择KAReadSourceStreamWriteSinkStream策略，然后选择附加策略.

现在，您已经创建了应用程序用来访问资源的服务执行角色。记下新角色的 ARN。

适用于step-by-step有关创建角色的说明，请参阅创建 IAM 角色 (控制台)中的IAM 用户指南.

创建 Kinesis Data Analytics 应用程序

1. 将以下 JSON 代码保存到名为 `create_request.json` 的文件中。将示例角色 ARN 替换为您之前创建的角色的 ARN。将存储桶 ARN 后缀 (*`username`*) 替换为在前一部分中选择的后缀。将服务执行角色中的示例账户 ID (*`012345678901`*) 替换为您的账户 ID。

```
{
    "ApplicationName": "test",
```

```
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_8",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/KA-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
    }
}
```

2. 使用上述请求执行 `CreateApplication` 操作来创建应用程序：

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

应用程序现已创建。您在下一步中启动应用程序。

启动应用程序

在本节中，您使用 `StartApplication` 操作来启动应用程序。

启动应用程序

1. 将以下 JSON 代码保存到名为 `start_request.json` 的文件中。

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. 使用上述请求执行 `StartApplication` 操作来启动应用程序：

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

应用程序正在运行。您可以在亚马逊上查看 Kinesis Data Analytics 指标CloudWatch控制台以验证应用程序是否正常运行。

停止应用程序

在本节中，您使用 `StopApplication` 操作来停止应用程序。

停止应用程序

1. 将以下 JSON 代码保存到名为 `stop_request.json` 的文件中。

```
{
   "ApplicationName": "test"
}
```

2. 使用下面的请求执行 `StopApplication` 操作来停止应用程序：

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

应用程序现已停止。

添加 CloudWatch 日志记录选项

您可以使用Amazon CLI添加亚马逊CloudWatch将流记录到您的应用程序。有关使用的信息CloudWatch在应用程序中记录，请参阅the section called “设置日志记录” (p. 205).

更新环境属性

在本节中，您使用 `UpdateApplication` 操作更改应用程序的环境属性，而无需重新编译应用程序代码。在本示例中，您更改源流和目标流的区域。

更新应用程序的环境属性

1. 将以下 JSON 代码保存到名为 `update_properties_request.json` 的文件中。

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. 使用前面的请求执行 `UpdateApplication` 操作以更新环境属性：

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

更新应用程序代码

在您需要使用新版本的代码包更新应用程序代码时，您可以使用 `UpdateApplication` Amazon CLI 操作。

> Note
>
> 要使用相同的文件名加载新版本的应用程序代码，您必须指定新的对象版本。有关使用 Amazon S3 对象版本的更多信息，请参阅启用或禁用版本控制.

使用Amazon CLI，请从 Amazon S3 存储桶中删除以前的代码包，上传新版本，然后调用并调用`UpdateApplication`指定相同的 Amazon S3 存储桶和对象名称以及新的对象版本。应用程序将使用新的代码包重新启动。

以下示例 `UpdateApplication` 操作请求重新加载应用程序代码并重新启动应用程序。将 `CurrentApplicationVersionId` 更新为当前的应用程序版本。您可以使用 `ListApplications` 或 `DescribeApplication` 操作检查当前的应用程序版本。将存储桶名称后缀 (*<username>*) 更新为在the section called “创建两个 Amazon Kinesis Data Streams” (p. 281)一节中选择的后缀。

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

下一个步骤

第 4 步：清理 Amazon 资源 (p. 291)

第 4 步：清理 Amazon 资源

本节包含清理在入门教程中创建的 Amazon 资源的过程。

本主题包含下列部分：

- 删除 Kinesis Data Analytics 应用程序 (p. 291)
- 删除 Kinesis Data Streams (p. 292)
- 删除您的 Amazon S3 对象和存储桶 (p. 292)
- 删除 IAM 资源 (p. 292)
- 删除 CloudWatch 资源 (p. 292)

删除 Kinesis Data Analytics 应用程序

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

2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 选择 Configure（配置）。
4. 在 Snapshots (快照) 部分中，选择 Disable (禁用)，然后选择 Update (更新)。
5. 在应用程序的页面中，选择 Delete (删除)，然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Streams 面板中，选择ExampleInput流.
3. 在ExampleInput流页面上，选择删除 Kinesis Streams然后确认删除。
4. 在Kinesis Streams页面上，选择ExampleOutput流，选择操作，选择Delete，然后确认删除。

删除您的 Amazon S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择ka-app-code-***<username>***存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除 IAM 资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-***<your-region>***政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏中，选择 Roles（角色）。
7. 选择Kinesis-Analytics-MyApplication-***<your-region>***角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除 CloudWatch 资源

1. 打开 CloudWatch 控制台 (https://console.aws.amazon.com/cloudwatch/)。
2. 在导航栏中，选择 Logs (日志)。
3. 选择/aws/kinesis-析ics/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

入门：Flink 1.6.2

本主题包含的入门 (DataStreamAPI) (p. 76)使用 Apache Flink 1.6.2 的教程。

主题

Apache Flink 应用程序的 Kinesis Data Analytics 应用程序的组件

要处理数据，您的 Kinesis Data Analytics 应用程序使用 Java/Apache Maven 或 Scala 应用程序，该应用程序使用 Apache Flink 运行时处理输入和生成输出。

Apache Flink Kinesis Data Analytics 应用程序具有以下组件：

- 运行时属性：您可以使用运行时属性配置应用程序，而无需重新编译应用程序代码。
- 源：应用程序通过使用资源. 源连接器从 Kinesis Data Streams、Amazon S3 存储桶等读取数据。有关更多信息，请参阅 源 (p. 8)。
- 运算符：应用程序使用一个或多个来处理数据。运营商. 操作符可以转换、丰富或聚合数据。有关更多信息，请参阅 DataStreamAPI 操作符 (p. 12)。
- 接收器：该应用程序使用以下方法为外部源生成数水槽. 接收器连接器将数据写入 Kinesis Data Fire 传输流、Amazon S3 存储桶等。有关更多信息，请参阅 接收器 (p. 9)。

创建、编译和打包应用程序后，您将代码包上传到 Amazon Simple Storage Service (Amazon S3) 存储桶。然后，您创建 Kinesis Data Analytics 应用程序。您在代码包位置中传入 Kinesis 数据流以作为流数据源，它通常是接收应用程序处理的数据的流或文件位置。

完成练习的先决条件

要完成本指南中的步骤，您必须满足以下条件：

- Java 开发工具包 (JDK) 版本 8。设置 `JAVA_HOME` 环境变量，使其指向您的 JDK 安装位置。
- 我们建议您使用开发环境（如 Eclipse Java Neon 或 IntelliJ Idea）来开发和编译您的应用程序。
- Git 客户端。如果尚未安装 Git 客户端，请安装它。
- Apache Maven 编译器插件。Maven 必须位于您的有效路径中。要测试您的 Apache Maven 安装，请输入以下内容：

```
$ mvn -version
```

要开始，请转到第 1 步：设置Amazon帐户并创建管理员用户 (p. 293)。

第 1 步：设置Amazon帐户并创建管理员用户

首次使用 Kinesis Data Analytics 前，请完成以下任务：

1. 注册 Amazon (p. 293)
2. 创建 IAM 用户 (p. 294)

注册 Amazon

当你注册时Amazon，您的Amazon账户会自动注册以获得中的所有服务Amazon，包括 Kinesis Data Analytics 您仅需为实际使用的服务付费。

使用 Kinesis Data Analytics 时，您只需为实际使用的资源付费。如果您是新手Amazon客户，您可以免费试用 Kinesis Data Analytics。有关更多信息，请参阅 Amazon 免费套餐。

如果您已有 Amazon 账户，请跳到下一个任务。如果您没有Amazon请执行以下步骤创建账户。

创建 Amazon 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。

注意您的Amazon账户 ID 是因为进行下一个任务时需要用到。

创建 IAM 用户

在中的服务AmazonKinesis Data Analytics (如) 要求您在访问时提供凭证。这样，服务才能确定您是否有权访问该服务所拥有的资源。Amazon Web Services Management Console要求您输入密码。

您可以为您的Amazon账户以访问Amazon Command Line Interface(Amazon CLI) 或者 API。但是，我们不建议使用您的Amazon账户的凭证访问Amazon。相反，我们建议您使用 Amazon Identity and Access Management (IAM)。创建 IAM 用户，将该用户添加到具有管理权限的 IAM 组，然后向您创建的 IAM 用户授予管理权限。您随后便可以使用一个特殊的 URL 和该 IAM 用户的凭证访问 Amazon。

如果你注册了Amazon，但您尚未为自己创建 IAM 用户，您可以使用 IAM 控制台自行创建。

本指南中的入门练习假定您拥有具有管理员权限的用户 (`adminuser`)。请按照以下过程在您的账户中创建 `adminuser`。

为管理员创建组

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Groups (组)，然后选择 Create New Group (创建新组)。
3. 对于组名，输入组的名称，例如 **`Administrators`**，然后选择 下一步。
4. 在策略列表中，选中AdministratorAccess政策。您可以使用 Filter (筛选) 菜单和 Search (搜索) 框来筛选策略列表。
5. 选择 Next Step (下一步)，然后选择 Create Group (创建组)。

您的新组列在 Group Name 下方。

要为您自己创建 IAM 用户，请将其添加到 Administrator 组中，然后创建密码

1. 在导航窗格中，选择 Users，然后选择 Add user。
2. 在 User name(用户名) 框中，输入一个用户名。
3. 选择这两者以编程方式访问和Amazon管理控制台访问.
4. 选择 Next:。Permissions (下一步：权限)。
5. 选中 Administrators 组旁的复选框。接下来，选择 Next (下一步)：审核。
6. 选择 Create user。

以新 IAM 用户身份登录

1. 注销 Amazon Web Services Management Console。

2. 使用下面的 URL 格式登录控制台：

   `https://aws_account_number.signin.aws.amazon.com/console/`

   这些区域有：*aws_account_number*是你的Amazon没有连字符的账户 ID。例如，如果您的Amazon账户 ID 是 1234-5678-9012，替换*aws_account_number*和**123456789012**. 有关如何查找您的账户号码的更多信息，请参阅。您的Amazon账户 ID 及其别名中的IAM 用户指南.
3. 输入您刚创建的 IAM 用户名和密码。登录后，导航栏将显示 *your_user_name* @ *your_aws_account_id*。

   Note

   如果您不希望您的登录页面 URL 包含 Amazon 账户 ID，可以创建账户别名。

创建或删除账户别名

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格上，选择 Dashboard。
3. 查找 IAM 用户登录链接。
4. 要创建别名，请选择 Customize (自定义)。输入要用于别名的名称，然后选择 Yes, Create (是，创建)。
5. 要删除别名，请选择 Customize，然后选择 Yes, Delete。登录 URL 会恢复使用 Amazon 账户 ID。

要在创建账户别名后登录，请使用以下 URL：

`https://your_account_alias.signin.aws.amazon.com/console/`

要为您的账户验证 IAM 用户的登录链接，请打开 IAM 控制台并在控制面板的 IAM 用户登录链接下进行检查。

有关 IAM 的更多信息，请参阅以下文档：

- Amazon Identity and Access Management (IAM)
- 入门
- IAM 用户指南

下一个步骤

第 2 步：设置 Amazon Command Line Interface (Amazon CLI) (p. 295)

第 2 步：设置 Amazon Command Line Interface (Amazon CLI)

在此步骤中，您将下载并配置Amazon CLI要与 Kinesis Data Analytics 结合使用 Apache Flink。

Note

本指南中的入门练习假定您使用账户中的管理员凭证 (`adminuser`) 来执行这些操作。

Note

如果您已安装 Amazon CLI，您可能需要升级以获得最新的功能。有关更多信息，请参阅 。安装 Amazon Command Line Interface中的Amazon Command Line Interface用户指南. 要检查 Amazon CLI 的版本，请运行以下命令：

```
aws --version
```

本教程中的练习需要以下 Amazon CLI 版本或更高版本：

```
aws-cli/1.16.63
```

设置 Amazon CLI

1. 下载并配置 Amazon CLI。有关说明，请参阅《Amazon Command Line Interface 用户指南》中的以下主题：
   - 安装 Amazon Command Line Interface
   - 配置 Amazon CLI
2. 在 Amazon CLI `config` 文件中为管理员用户添加一个命名的配置文件。在执行 Amazon CLI 命令时，您将使用此配置文件。有关指定配置文件的更多信息，请参命名配置文件中的Amazon Command Line Interface用户指南.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

   有关可用的列表Amazon地区，请参阅区域和终端节点中的Amazon Web Services 一般参考.

   > **Note**
   >
   > 本教程中的示例代码和命令使用美国西部（俄勒冈）区域。要使用不同的区域，请将本教程的代码和命令中的区域更改为要使用的区域。

3. 在命令提示符处输入以下帮助命令来验证设置：

   ```
   aws help
   ```

在你设置之后Amazon账户和Amazon CLI中，您可以尝试下一个练习中，在其中，您将配置一个示例应用程序并测试end-to-end设置。

下一个步骤

第 3 步：创建和运行 Kinesis Data Analytics 应用程序 (p. 296)

第 3 步：创建和运行 Kinesis Data Analytics 应用程序

在本练习中，您创建 Kinesis Data Analytics 应用程序，并将数据流作为源和接收器。

本节包含以下步骤：

创建两个 Amazon Kinesis Data Streams

在为本练习创建 Kinesis Data Analytics 应用程序前，请创建两个 Kinesis Data Analytics 应用程序 (`ExampleInputStream`和`ExampleOutputStream`)。您的应用程序将这些数据流用于应用程序源和目标流。

您可以使用 Amazon Kinesis 控制台或以下内容来创建这些流：Amazon CLI命令。有关控制台说明，请参阅创建和更新数据流中的Amazon Kinesis Data Streams 开发人员指南.

创建数据流 (Amazon CLI)

1. 创建第一个流 (`ExampleInputStream`)，请使用以下 Amazon Kinesis`create-stream` Amazon CLI 命令。

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. 要创建应用程序用来写入输出的第二个流，请运行同一命令（将流名称更改为 `ExampleOutputStream`）。

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

将示例记录写入输入流

在本节中，您使用 Python 脚本将示例记录写入流，以供应用程序处理。

> **Note**
>
> 此部分需要 Amazon SDK for Python (Boto)。

1. 使用以下内容创建名为 `stock.py` 的文件：

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
```

```
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

2. 在本教程的后面部分，您运行 `stock.py` 脚本，以将数据发送到应用程序。

```
$ python stock.py
```

下载并检查 Apache Flink 流式处理 Java 代码

此示例的 Java 应用程序代码可从以下网址获得。GitHub. 要下载应用程序代码，请执行以下操作：

1. 使用以下命令克隆远程存储库：

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-java-examples
```

2. 导航到 `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_6`目录。

请注意有关应用程序代码的以下信息：

- 一个项目对象模型 (pom.xml)文件包含有关应用程序的配置和依赖关系的信息，包括 Apache Flink 库的 Kinesis Data Analytics 库。
- `BasicStreamingJob.java` 文件包含定义应用程序功能的 `main` 方法。
- 该应用程序使用 Kinesis 源从源流中进行读取。以下代码段创建 Kinesis 源：

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- 您的应用程序使用 `StreamExecutionEnvironment` 对象创建源和接收连接器以访问外部资源。
- 该应用程序将使用静态属性创建源和接收连接器。要使用动态应用程序属性，请使用 `createSourceFromApplicationProperties` 和 `createSinkFromApplicationProperties` 方法以创建连接器。这些方法读取应用程序的属性来配置连接器。

  有关运行时属性的更多信息，请参阅运行时属性 (p. 18)。

编译应用程序代码

在本节中，您使用 Apache Maven 编译器创建应用程序的 Java 代码。有关安装 Apache Maven 和 Java 开发工具包 (JDK) 的信息，请参阅完成练习的先决条件 (p. 293)。

> **Note**
>
> 要将 Kinesis 连接器与 1.11 以前的 Apache Flink 版本一起使用，您需要下载连接器源代码并构建该连接器，如Apache Flink 文档.

编译应用程序代码

1. 要使用您的应用程序代码，您将其编译和打包成 JAR 文件。您可以通过两种方式之一编译和打包您的代码：

- 使用命令行 Maven 工具。在包含 `pom.xml` 文件的目录中通过运行以下命令创建您的 JAR 文件：

```
mvn package
```

Note

Kinesis Data Analytics 运行时版本 1.0.1 不需要使用-dflink.version 参数；仅 1.1.0 及更高版本需要使用该参数。有关更多信息，请参阅 the section called “指定应用程序的 Apache Flink 版本” (p. 4)。

- 设置开发环境。有关详细信息，请参阅您的开发环境文档。

您可以作为 JAR 文件上传您的包，也可以将包压缩为 ZIP 文件并上传。如果您使用 Amazon CLI 创建应用程序，您可以指定您的代码内容类型（JAR 或 ZIP）。

2. 如果编译时出错，请验证 `JAVA_HOME` 环境变量设置正确。

如果应用程序成功编译，则创建以下文件：

`target/aws-kinesis-analytics-java-apps-1.0.jar`

上传 Apache Flink 流式处理 Java 代码

在本节中，您创建 Amazon Simple Storage Service (Amazon S3) 存储桶并上传应用程序代码。

上传应用程序代码

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. Enter**`ka-app-code-<username>`**中的Bucket name字段中返回的子位置类型。将后缀（如您的用户名）添加到存储桶名称，以使其具有全局唯一性。请选择 Next (下一步)。
4. 在配置选项步骤中，让设置保持原样，然后选择下一步。
5. 在设置权限步骤中，让设置保持原样，然后选择下一步。
6. 请选择 Create bucket (创建存储桶)。
7. 在 Amazon S3 控制台中，选择ka-app-code-***`<username>`***存储桶，然后选择上传.
8. 在选择文件步骤中，选择添加文件。导航到您在上一步中创建的 `aws-kinesis-analytics-java-apps-1.0.jar` 文件。请选择 Next (下一步)。
9. 在设置权限步骤中，让设置保持原样。请选择 Next (下一步)。
10. 在设置属性步骤中，让设置保持原样。请选择 Upload（上传）。

您的应用程序代码现在存储在 Amazon S3 存储桶中，应用程序可以在其中访问代码。

创建和运行 Kinesis Data Analytics 应用程序

您可以使用控制台或创建和运行 Kinesis Data Analytics 应用程序。Amazon CLI.

Note

当您使用控制台创建应用程序时，您的Amazon Identity and Access Management(IAM) 和亚马逊CloudWatch日志资源是为你创建的。当您使用 Amazon CLI 创建应用程序时，您可以单独创建这些资源。

主题

创建并运行应用程序（控制台）

按照以下步骤，使用控制台创建、配置、更新和运行应用程序。

创建 应用程序

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics.
2. 在 Amazon Kinesis Data Analytics 仪表板中，选择创建分析应用.
3. 在 Kinesis Analytics – 创建应用程序页面上，提供应用程序详细信息，如下所示：
   - 对于 Application name (应用程序名称)，输入 **`MyApplication`**。
   - 对于描述，输入 **`My java test app`**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。

     Note

     Kinesis Data Analytics 使用 Apache Flink 版本 1.8.2 或 1.6.2。
   - 将版本下拉列表更改为 Apache Flink 1.6。
4. 对于访问权限，请选择 Create / update IAM role (创建/更新 IAM 角色) **`kinesis-analytics-MyApplication-us-west-2`**。
5. 选择 Create application（创建应用程序）。

   Note

   当您使用控制台创建 Kinesis Data Analytics 应用程序时，您可以选择为您的应用程序创建 IAM 角色和策略。您的应用程序使用此角色和策略访问其从属资源。这些 IAM 资源使用您的应用程序名称和区域命名，如下所示：

   - 策略：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
   - 角色：`kinesis-analytics-`*`MyApplication`*`-`*`us-west-2`*

编辑 IAM 策略

编辑 IAM 策略以添加访问 Kinesis Data Streams 的权限。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择策略。选择控制台在上一部分中为您创建的 **`kinesis-analytics-service-MyApplication-us-west-2`** 策略。
3. 在 Summary (摘要) 页面上，选择 Edit policy (编辑策略)。请选择 JSON 选项卡。
4. 将以下策略示例中突出显示的部分添加到策略中。将示例账户 ID (*`012345678901`*) 替换为您的账户 ID。

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

```
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/java-getting-started-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

配置应用程序

1. 在存储库的MyApplication页面上，选择配置.
2. 在 Configure application (配置应用程序) 页面上，提供 Code location (代码位置)：
   - 适用于Amazon S3 存储桶输入**ka-app-code-*<username>***.
   - 适用于Amazon S3 对象的路径输入**java-getting-started-1.0.jar**.

3. 在 Access to application resources (对应用程序的访问权限) 下，对于 Access permissions (访问权限)，选择 Create / update IAM role (创建/更新 IAM 角色) **kinesis-analytics-MyApplication-us-west-2**。
4. 在 Properties (属性) 下，对于 Group ID (组 ID)，输入 **ProducerConfigProperties**。
5. 输入以下应用程序属性和值：

密钥	值
flink.inputstream.initpos	**LATEST**
aws:region	**us-west-2**
AggregationEnabled	**false**

6. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
7. 适用于CloudWatch记录，选择启用”复选框。
8. 选择 Update（更新）。

Note

当你选择启用亚马逊CloudWatch日志记录中，Kinesis Data Analytics 为您创建日志组和日志流。这些资源的名称如下所示：

- 日志组：`/aws/kinesis-analytics/MyApplication`
- 日志流：`kinesis-analytics-log-stream`

运行应用程序

1. 在存储库的MyApplication页面上，选择运行. 确认该操作。
2. 当应用程序正在运行时，请刷新页面。控制台将显示 Application graph (应用程序图表)。

停止应用程序

在存储库的MyApplication页面上，选择停止. 确认该操作。

更新应用程序

使用控制台，您可以更新应用程序设置，例如应用程序属性、监控设置，或应用程序 JAR 文件的位置和文件名。如果您需要更新应用程序代码，您还可以从 Amazon S3 存储桶重新加载应用程序 JAR。

在存储库的MyApplication页面上，选择配置. 更新应用程序设置，然后选择更新。

创建并运行应用程序 (Amazon CLI)

在本部分中，您将使用Amazon CLI创建和运行 Kinesis Data Analytics 应用程序。Apache Flink 的 Kinesis Data Analytics 使用`kinesisanalyticsv2` Amazon CLI命令来创建 Kinesis Data Analytics 应用程序并与之交互。

创建权限策略

首先，使用两个语句创建权限策略：一个语句授予对源流执行 `read` 操作的权限，另一个语句授予对接收器流执行 `write` 操作的权限。然后，您将策略附加到 IAM 角色（下一部分中将创建此角色）。因此，在 Kinesis Data Analytics 代入该角色时，服务具有必要的权限从源流进行读取和写入接收器流。

使用以下代码创建 KAReadSourceStreamWriteSinkStream 权限策略。Replace*username*使用您用于创建 Amazon S3 存储桶以存储应用程序代码的用户名。将 Amazon 资源名称 (ARN) 中的账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleOutputStream"
        }
    ]
}
```

适用于step-by-step有关创建权限策略的说明，请参阅教程：创建并附加您的第一个客户托管策略中的IAM 用户指南.

> Note
>
> 要访问其他亚马逊服务，您可以使用Amazon SDK for Java. Kinesis Data Analytics 会自动将开发工具包所需的凭证设置为与您的应用程序关联的服务执行 IAM 角色的凭证。无需执行其他步骤。

创建 IAM 角色

在本节中，您创建一个 IAM 角色，Kinesis Data Analytics 应用程序可以代入此角色来读取源流并写入接收器流。

没有权限，Kinesis Data Analytics 将无法访问您的流。您通过 IAM 角色授予这些权限。每个 IAM 角色附加了两个策略。此信任策略授予 Kinesis Data Analytics 入该角色的权限，权限策略确定代入这个角色后可以执行的操作。

您将在上一部分中创建的权限策略附加到此角色。

创建 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles (角色) 和 Create Role (创建角色)。
3. UNDER选择受信任的身份类型，选择Amazon服务. 在 Choose the service that will use this role (选择将使用此角色的服务) 下，选择 Kinesis。在选择您的使用案例下，选择 Kinesis Analytics。

   选择 Next:。Permissions (下一步：权限)。
4. 在存储库的附加权限策略页面上，选择后续：审核。在创建角色后，您可以附加权限策略。

5. 在存储库的创建角色页面，输入**KA-stream-rw-role**(对于)Role name (角色名称). 请选择 Create role（创建角色）。

   现在，您已经创建了一个名为的新 IAM 角色KA-stream-rw-role. 接下来，您更新角色的信任和权限策略。

6. 将权限策略附加到角色。

   Note

   在本练习中，Kinesis Data Analytics 代入此角色，以便同时从 Kinesis 数据流（源）读取数据和将输出写入另一个 Kinesis 数据流。因此，您附加在上一步（the section called “创建权限策略” (p. 302)）中创建的策略。

   a. 在 Summary (摘要) 页上，选择 Permissions (权限) 选项卡。
   b. 选择附加策略。
   c. 在搜索框中，输入 **KAReadSourceStreamWriteSinkStream**（您在上一部分中创建的策略）。
   d. 选择KAReadSourceStreamWriteSinkStream策略，然后选择附加策略.

现在，您已经创建了应用程序用来访问资源的服务执行角色。记下新角色的 ARN。

适用于step-by-step有关创建角色的说明，请参阅创建 IAM 角色 (控制台)中的IAM 用户指南.

创建 Kinesis Data Analytics 应用程序

1. 将以下 JSON 代码保存到名为 `create_request.json` 的文件中。将示例角色 ARN 替换为您之前创建的角色的 ARN。将存储桶 ARN 后缀 (*username*) 替换为在前一部分中选择的后缀。将服务执行角色中的示例账户 ID (*012345678901*) 替换为您的账户 ID。

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_6",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/KA-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "java-getting-started-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
```

```
    }
}
```

2. 使用上述请求执行 CreateApplication 操作来创建应用程序：

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

应用程序现已创建。您在下一步中启动应用程序。

启动应用程序

在本节中，您使用 StartApplication 操作来启动应用程序。

启动应用程序

1. 将以下 JSON 代码保存到名为 start_request.json 的文件中。

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. 使用上述请求执行 StartApplication 操作来启动应用程序：

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

应用程序正在运行。您可以在亚马逊上查看 Kinesis Data Analytics 指标CloudWatch控制台以验证应用程序是否正常工作。

停止应用程序

在本节中，您使用 StopApplication 操作来停止应用程序。

停止应用程序

1. 将以下 JSON 代码保存到名为 stop_request.json 的文件中。

```
{
   "ApplicationName": "test"
}
```

2. 使用下面的请求执行 StopApplication 操作来停止应用程序：

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

应用程序现已停止。

添加 CloudWatch 日志记录选项

您可以使用Amazon CLI添加亚马逊CloudWatch将流记录到应用程序中。有关使用的信息CloudWatch登录应用程序，请参阅the section called “设置日志记录” (p. 205).

更新环境属性

在本节中，您使用 UpdateApplication 操作更改应用程序的环境属性，而无需重新编译应用程序代码。在本示例中，您更改源流和目标流的区域。

更新应用程序的环境属性

1. 将以下 JSON 代码保存到名为 update_properties_request.json 的文件中。

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                     "flink.stream.initpos" : "LATEST",
                     "aws.region" : "us-west-2",
                     "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                     "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. 使用前面的请求执行 UpdateApplication 操作以更新环境属性：

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

更新应用程序代码

在您需要使用新版本的代码包更新应用程序代码时，您可以使用 UpdateApplication Amazon CLI 操作。

使用Amazon CLI，请从 Amazon S3 存储桶中删除以前的代码包，上传新版本，然后调用并调用UpdateApplication指定相同的 Amazon S3 存储桶和对象名称。应用程序将使用新的代码包重新启动。

以下示例 UpdateApplication 操作请求重新加载应用程序代码并重新启动应用程序。将 CurrentApplicationVersionId 更新为当前的应用程序版本。您可以使用 ListApplications 或 DescribeApplication 操作检查当前的应用程序版本。将存储桶名称后缀 (*<username>*) 更新为在the section called “创建两个 Amazon Kinesis Data Streams” (p. 297)一节中选择的后缀。

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "java-getting-started-1.0.jar"
```

```
                }
            }
        }
    }
}
```

第 4 步：清理 Amazon 资源

本节包含清理在入门教程中创建的 Amazon 资源的过程。

本主题包含下列部分：

- 删除 Kinesis Data Analytics 应用程序 (p. 307)
- 删除 Kinesis Data Streams (p. 307)
- 删除您的 Amazon S3 对象和存储桶 (p. 307)
- 删除 IAM 资源 (p. 307)
- 删除 CloudWatch 资源 (p. 308)

删除 Kinesis Data Analytics 应用程序

1. 打开 Kinesis 控制台，网址为：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Analytics 面板中，选择MyApplication.
3. 选择 Configure（配置）。
4. 在 Snapshots (快照) 部分中，选择 Disable (禁用)，然后选择 Update (更新)。
5. 在应用程序的页面中，选择 Delete (删除)，然后确认删除。

删除 Kinesis Data Streams

1. 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics.
2. 在 Kinesis Data Streams 面板中，选择ExampleInput流.
3. 在ExampleInput流页面上，选择删除 Kinesis Streams然后确认删除。
4. 在Kinesis Streams页面上，选择ExampleOutput流，选择操作，选择Delete，然后确认删除。

删除您的 Amazon S3 对象和存储桶

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择ka-app-code-***<username>***存储桶。
3. 选择 Delete (删除)，然后输入存储桶名称以确认删除。

删除 IAM 资源

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航栏中，选择策略。
3. 在筛选条件控件中，输入 kinesis。
4. 选择kinesis-analytics-service-MyApplication-***<your-region>***政策。
5. 选择 Policy Actions (策略操作)，然后选择 Delete (删除)。
6. 在导航栏中，选择 Roles（角色）。

7. 选择kinesis-analytics-MyApplication-*<your-region>*角色。
8. 选择 Delete role (删除角色)，然后确认删除。

删除 CloudWatch 资源

1. 打开 CloudWatch 控制台 (https://console.aws.amazon.com/cloudwatch/)。
2. 在导航栏中，选择 Logs (日志)。
3. 选择/aws/kinesis-analytics/MyApplication日志组。
4. 选择 Delete Log Group (删除日志组)，然后确认删除。

Apache Flink 设置

Apache Flink 的 Kinesis Data Analytics 是 Apache Flink 框架的实施形式。Kinesis Data Analytics 使用本节中所述的默认值。其中一些值可以由 Kinesis Data Analytics 应用程序在代码中设置，但无法更改其他值。

本主题包含下列部分：

Apache Flink 配置

Kinesis Data Analytics 提供了默认的 Flink 配置，其中包括 Apache Flink 推荐的大多数属性值以及一些基于常见应用程序配置文件的值。有关 Flink 配置的更多信息，请参阅配置. 服务提供的默认配置适用于大多数应用。但是，如果您需要调整 Flink 配置属性以提高某些具有高并行度、内存和状态使用率高的应用程序的性能，或者在 Apache Flink 中启用新的调试功能，则可以通过请求支持案例来更改某些属性。有关更多信息，请参阅 Amazon 支持中心。您可以使用Apache Flink 控制面板.

状态后端

Kinesis Data Analytics 将临时数据存储在状态后端中。Kinesis Data Analytics 使用RockSDBStateBackend. 调用 `setStateBackend` 以设置不同的后端无效。

我们在状态后端上启用以下功能：

- 增量状态后端快照
- 异步状态后端快照
- 本地检查点恢复

在 Kinesis Data Analytics 中，`state.backend.rocksdb.ttl.compaction.filter.enabled`默认情况下，配置已启用。通过使用该筛选条件，您可以更新应用程序代码以启用压缩清理策略。有关更多信息，请参阅 。Flink 中的州 TTL中的Apache Flink 文档.

有关状态后端的更多信息，请参阅状态后端中的Apache Flink 文档.

检查点

Apache Flink 的 Kinesis Data Analytics 使用具有以下值的默认检查点配置。可以更改其中的一些值。你必须设置CheckpointConfiguration.ConfigurationType到`CUSTOM`为 Kinesis Data Analytics 使用修改的检查点值。

设置	是否可以修改？	默认值
CheckpointingEnabled	可修改	True

设置	是否可以修改？	默认值
CheckpointInterval	可修改	60000
MinPauseBetweenCheckpoints	可修改	5000
并发检查点数	不能修改	1
检查点模式	不能修改	恰好一次
检查点保留策略	不能修改	失败时
检查点超时	不能修改	60 分钟
保留的最大检查点数	不能修改	1
重新启动策略	不能修改	固定延迟，每 10 秒无限次重试。
检查点和保存点位置	不能修改	我们将持久的检查点和保存点数据存储到服务拥有的 S3 存储桶中。
状态后端内存阈值	不能修改	1048576

保存点

默认情况下，从保存点中还原时，恢复操作尝试将保存点的所有状态映回到用于还原的程序。如果删除了一个操作符，默认情况下，从包含与缺少的操作符对应的数据的保存点中还原将失败。您可以通过设置AllowNonRestoredState应用程序的参数FlinkRun配置到`true`. 这样，恢复操作就可以跳过无法映射到新程序的状态。

有关更多信息，请参阅 。允许未恢复状态中的Apache Flink 文档.

堆大小

Kinesis Data Analytics 为每个 KPU 分配 3 GiB JVM 堆，并为本机代码分配保留 1 GiB。有关增加应用程序容量的信息，请参阅the section called “扩缩” (p. 27)。

有关 JVM 堆大小的更多信息，请参阅配置中的Apache Flink 文档.

为 Apache Flink 配置 Kinesis Data Analytics 以访问 Amazon VPC 中的资源

您可以配置 Kinesis Data Analytics 应用程序以连接到您账户的 Virtual Private Cloud (VPC) 中的私有子网。使用 Amazon Virtual Private Cloud (Amazon VPC) 为资源（如数据库、缓存实例或内部服务）创建私有网络。将应用程序连接到 VPC 以在执行期间访问私有资源。

本主题包含下列部分：

Amazon VPC 概念

Amazon VPC 是 Amazon EC2 的网络层。如果您不熟悉 Amazon EC2，请参阅适用于 Linux 实例的 Amazon EC2 用户指南 中的什么是 Amazon EC2？以了解简要说明。

以下是 VPC 的主要概念：

- virtual private cloud (VPC) 是仅适用于您的Amazon账户的虚拟网络。
- 子网是您的 VPC 内的 IP 地址范围。
- 路由表 包含一组称为“路由”的规则，它们用于确定将网络流量发送到何处。
- Internet 网关 是一种横向扩展、冗余且高度可用的 VPC 组件，支持在 VPC 中的实例和 Internet 之间进行通信。因此它不会对网络流量造成可用性风险或带宽限制。
- 一个VPC 终端节点使您可以私下地将 VPC 连接到支持的Amazon支持的服务和 VPC 终端节点服务 PrivateLink无需互联网网关、NAT 设备、VPN 连接或Amazon 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 Amazon VPC 服务的更多信息，请参阅Amazon Virtual Private Cloud 用户指南.

Kinesis Data Analytics弹性网络接口在应用程序的 VPC 配置中提供的其中一个子网中。在 VPC 子网中创建的弹性网络接口数可能会有所不同，具体取决于应用程序的并行度和每个 KPU 的并行度。有关应用程序扩展的更多信息，请参阅扩缩 (p. 27)。

> Note
>
> SQL 应用程序不支持 VPC 配置。

> Note
>
> Kinesis Data Analytics 服务管理具有 VPC 配置的应用程序的检查点和快照状态。

VPC 应用程序权限

本节介绍了应用程序与 VPC 一起使用时所需的权限策略。有关使用权限策略的更多信息，请参阅 Identity and Access Management (p. 198)。

以下权限策略为应用程序授予与 VPC 交互所需的权限。要使用该权限策略，请将其添加到应用程序的执行角色中。

用于访问 Amazon VPC 的权限策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VPCReadOnlyPermissions",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeDhcpOptions"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ENIReadWritePermissions",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:CreateNetworkInterfacePermission",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DeleteNetworkInterface"
      ],
      "Resource": "*"
    }

    ]
}
```

Note

使用控制台指定应用程序资源时（例如CloudWatch控制台修改应用程序执行角色以授予访问这些资源的权限。只有在不使用控制台创建应用程序时，您才需要手动修改应用程序的执行角色。

VPC 连接的 Kinesis Data Analytics 应用程序的 Internet 和服务访问

默认情况下，在将 Kinesis Data Analytics 应用程序连接到您账户中的 VPC 时，除非 VPC 提供访问权限，否则它将无法访问 Internet。如果应用程序需要访问 Internet，则需要满足以下条件：

- 只能为使用私有子网 Kinesis Data Analytics 应用程序。
- VPC 必须在公有子网中包含 NAT 网关或实例。
- 出站流量必须具有从私有子网到公有子网中的 NAT 网关的路由。

Note

一些服务提供了 VPC 终端节点。您可以使用 VPC 终端节点从 VPC 内连接到 Amazon 服务，而无需 Internet 访问权限。

子网是公有还是私有子网取决于其路由表。每个路由表具有一个默认路由，它确定具有公有目标的数据包的下一跃点。

- 对于私有子网：默认路由指向 NAT 网关 (nat-...) 或 NAT 实例 (eni-...)。
- 对于公有子网：默认路由指向 Internet 网关 (igw-...)。

在为 VPC 配置一个公有子网（具有 NAT）以及一个或多个私有子网后，请执行以下操作以标识私有子网和公有子网：

- 在 VPC 控制台的导航窗格中，选择 Subnets (子网)。
- 选择一个子网，然后选择 Route Table (路由表) 选项卡。验证默认路由：
  - 公有子网：目标：0.0.0.0/0，目标：igw-...
  - 私有子网：目标：0.0.0.0/0，目标：nat-... 或 eni-... 或 eni-...

要将 Kinesis Data Analytics 应用程序与私有子网关联，请执

- 打开 Kinesis Data Analytics 控制台https://console.aws.amazon.com/kinesisanalytics.
- 在 Kinesis Analytics applications (Kinesis Analytics 应用程序) 页面上，选择您的应用程序，然后选择 Application details (应用程序详细信息)。
- 在应用程序页面上，选择 Configure (配置)。
- 在 VPC Connectivity (VPC 连接) 部分中，选择要与您的应用程序关联的 VPC。选择与您的 VPC 关联的子网和安全组，您希望应用程序使用它们访问 VPC 资源。
- 选择 Update（更新）。

相关信息

创建具有公有和私有子网的 VPC

NAT 网关基础知识

Kinesis Data Analytics VPC API

可以使用以下 Kinesis Data Analytics API 操作管理应用程序的 VPC。有关使用 Kinesis Data Analytics API 的信息，请参阅API 示例代码 (p. 341).

CreateApplication

使用CreateApplication在创建期间将 VPC 配置添加到应用程序中的操作。

`CreateApplication` 操作的以下示例请求代码在创建应用程序时包括 VPC 配置：

```
{
  "ApplicationName":"MyApplication",
  "ApplicationDescription":"My-Application-Description",
  "RuntimeEnvironment":"FLINK-1_13",
  "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
```

```
    "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType":"ZIPFILE"
    },
      "FlinkApplicationConfiguration":{
      "ParallelismConfiguration":{
        "ConfigurationType":"CUSTOM",
        "Parallelism":2,
        "ParallelismPerKPU":1,
        "AutoScalingEnabled":true
      }
    },
  "VpcConfigurations": [
        {
           "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
           "SubnetIds": [ "subnet-0123456789abcdef0" ]
        }
     ]
  }
}
```

AddApplicationVpcConfiguration

使用AddApplicationVpcConfiguration在创建应用程序后，可以使用操作将 VPC 配置添加到应用程序中。

`AddApplicationVpcConfiguration` 操作的以下示例请求代码将 VPC 配置添加到现有应用程序中：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
   }
}
```

DeleteApplicationVpcConfiguration

使用DeleteApplicationVpcConfiguration从应用程序中删除 VPC 配置的操作。

`AddApplicationVpcConfiguration` 操作的以下示例请求代码从应用程序中删除现有的 VPC 配置：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfigurationId": "1.1"
}
```

UpdateApplication

使用UpdateApplication可以使用操作同时更新应用程序的所有 VPC 配置。

`UpdateApplication` 操作的以下示例请求代码更新应用程序的所有 VPC 配置：

```
{
   "ApplicationConfigurationUpdate": {
      "VpcConfigurationUpdates": [
         {
            "SecurityGroupIdUpdates": [ "sg-0123456789abcdef0" ],
            "SubnetIdUpdates": [ "subnet-0123456789abcdef0" ],
            "VpcConfigurationId": "2.1"
         }
      ]
   },
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9
}
```

例如：使用 VPC 访问 Amazon MSK 集群中的数据

有关如何从 VPC 上的 Amazon MSK 集群中访问数据的完整教程，请参阅 MSK 复制 (p. 145)。

问题 Kinesis Data Analytics，Apache Flink

以下内容可帮助您解决在 Amazon Kinesis Data Analytics（适用于 Apache Flink）中可能遇到的问题。

主题

开发排查

主题

Apache Flink 火焰图

默认情况下，在支持它的 Apache Flink 版本的 Kinesis Data Analytics 中的应用程序上启用火焰图。如果保持图表打开状态，火焰图可能会影响应用程序性能，如中所述Flink 文档.

如果要为应用程序禁用 Flame Graph，请创建一个案例以请求为应用程序 ARN 禁用它。有关更多信息，请参阅 。Amazon支持中心.

问题是 EFO 连接器 1.13.2

1.13.2 Kinesis Data Streams EFO 连接器存在一个已知问题，如果应用程序受到高背压，则性能会降低。为了缓解问题，请使用 Flink 1.13.3 连接器。您可以使用1.13-快照在 Maven 上。

编译错误：“无法解析项目的依赖项”

要编译 Apache Flink 示例应用程序的 Kinesis Data Analytics，您必须先下载并编译 Apache Flink Kinesis 连接器，然后将其添加到本地 Maven 存储库中。如果尚未将连接器添加到存储库中，则会显示类似下面的编译错误：

```
Could not resolve dependencies for project your project name: Failure to find
 org.apache.flink:flink-connector-kinesis_2.11:jar:1.8.2 in https://repo.maven.apache.org/
maven2 was cached in the local repository, resolution will not be reattempted until the
 update interval of central has elapsed or updates are forced
```

要解决该错误，您必须下载连接器的 Apache Flink 源代码（位于 https://flink.apache.org/downloads.html 中的版本 1.8.2）。有关如何下载、编译和安装 Apache Flink 源代码的说明，请参阅the section called “将 Apache Flink Kinesis Streams 连接器与早期 Apache Flink 版本一起使用” (p. 258)。

无效的选项：“kinesisanalyticsv2”

要使用 Kinesis Data Analytics API 的第 2 版，您需要安装Amazon Command Line Interface(Amazon CLI)。

有关升级Amazon CLI，请参阅安装Amazon Command Line Interface中的Amazon Command Line Interface用户指南.

UpdateApplication 操作没有重新加载应用程序代码

这些区域有：UpdateApplication如果未指定 S3 对象版本，操作不会使用相同的文件名重新加载应用程序代码。要使用相同的文件名重新加载应用程序代码，请在 S3 存储桶上启用版本控制，并使用`ObjectVersionUpdate`参数。有关在 S3 存储桶中启用对象版本控制的更多信息，请参阅启用或禁用版本控制。

运行时钟偏差

本节包含有关诊断和修复 Kinesis Data Analytics 应用程序运行时问题的信息。

主题

故障排除工具

检测应用程序问题的主要工具是 CloudWatch 警报。使用 CloudWatch 警报，您可以设置阈值 CloudWatch 指示应用程序中存在错误或瓶颈情况的指标。有关推荐的信息 CloudWatch 警报，请参阅使用 CloudWatch 使用 Amazon Kinesis Data Analytics 发出警报 (p. 226).

应用程序问题

本节包含针对 Kinesis Data Analytics 应用程序可能遇到的错误情况的解决方案。

主题

应用程序停滞在暂时状态

如果您的应用程序处于暂时状态 (STARTING、UPDATING、STOPPING，或者AUTOSCALING)，您可以使用停止您的应用程序StopApplication动作Forceparameter settrue. 你不能强制停止中的应用程序DELETINGstatus 或者，如果应用程序位于UPDATING要么AUTOSCALINGstatus，则可以将其回滚到之前运行的版本。回滚应用程序时，它会从上次成功的快照中加载状态数据。如果应用程序没有快照，Kinesis Data Analytics 会拒绝回滚请求。有关还原应用程序的更多信息，请参阅RollbackApplicationaction.

> Note
>
> 强制停止应用程序可能会导致数据丢失或重复。为防止应用程序重启期间数据丢失或重复处理数据，我们建议您频繁拍摄应用程序快照。

应用程序停滞的原因包括：

- 应用程序状态过大：如果应用程序状态过大或过于持久，可能会导致应用程序在执行检查点或快照操作期间停滞。检查应用程序的lastCheckpointDuration和lastCheckpointSize值稳步增加或值异常高的指标。
- 应用程序代码过大：确认应用程序 JAR 文件小于 512 MB。不支持超过 512 MB 的 JAR 文件。
- 应用程序快照创建失败：Kinesis Data Analytics 在应用程序的第一个版本期间拍摄应用程序的快照UpdateApplication要么StopApplication请求. 然后，该服务使用此快照状态，并使用更新的应用程序配置还原应用程序以提供恰好一次处理语义。如果自动快照创建失败，请参阅快照创建失败 (p. 318)以下。
- 从快照还原失败：如果在应用程序更新中删除或更改一个操作符并尝试从快照中还原，默认情况下，如果快照包含缺少的操作符的状态数据，还原将失败。此外，应用程序将停滞在任一STOPPED要么UPDATINGstatus 要更改此行为并允许恢复成功，请将AllowNonRestoredState应用程序的参数FlinkRunConfiguration到true. 这样，恢复操作就可以跳过无法映射到新程序的状态数据。
- 应用程序初始化需要更长的时间：Kinesis Data Analytics 在等待 Flink 作业启动时使用 5 分钟的内部超时时间（软设置）。如果您的作业未能在此超时时间内启动，您将看到 CloudWatch 日志如下所示：

```
Flink job did not start within a total timeout of 5 minutes for application: %s under
 account: %s
```

如果你遇到上述错误，则意味着你的操作在 Flink 作业的定义下main方法耗时超过 5 分钟，导致 Kinesis Data Analytics 端的 Flink 作业创建超时。我们建议你查看 FlinkJobManager日志以及您的应用程序代码，以查看此延迟是否在main方法是预期方法。如果没有，则需要采取措施解决问题，以便在不到5分钟的时间内完成。

您可以使用以下任一方式检查应用程序状态ListApplications或者DescribeApplication行动。

快照创建失败

在以下情况下，Kinesis Data Analytics 服务无法拍摄快照：

- 应用程序超过快照限制。快照限制为 1,000 个。有关更多信息，请参阅 快照 (p. 24)。
- 应用程序无权访问其源或接收器。
- 应用程序代码无法正常工作。
- 应用程序遇到其他配置问题。

如果在应用程序更新期间拍摄快照或停止应用程序时遇到异常，请设置`SnapshotsEnabled`你的应用程序的财产`ApplicationSnapshotConfiguration`到`false`然后重试该请求。

如果未正确配置应用程序的操作员，快照可能会失败。有关调整操作员性能的信息，请参阅运营商扩展 (p. 239).

在应用程序恢复正常状态后，我们建议您将应用程序的`SnapshotsEnabled`财产给`true`.

无法访问 VPC 中的资源

如果应用程序使用 Amazon VPC 上运行的 VPC，请执行以下操作以验证应用程序是否有权访问其资源：

- 检查您的 CloudWatch 记录以下错误。该错误表明应用程序无法访问 VPC 中的资源：

```
org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after 60000
 ms.
```

如果看到该错误，请确认正确设置了路由表，并且连接器具有正确的连接设置。

有关设置和分析的信息 CloudWatch 日志，请参阅日志记录和监控 (p. 204).

在写入 Amazon S3 存储桶时丢失数据

在使用 Apache Flink 版本 1.6.2 将输出写入到 Amazon S3 存储桶时，可能会丢失一些数据。在直接使用 Amazon S3 输出时，我们建议使用支持的最新 Apache Flink 版本。要使用 Apache Flink 1.6.2 写入到 Amazon S3 存储桶，我们建议使用 Kinesis Data Firehose。有关将 Kinesis Data Firehose 用于Kinesis Data Analytics 的更多信息，请参阅Kinesis Data Firehose (p. 155).

应用程序处于 RUNNING 状态但未处理数据

您可以使用以下任一选项检查应用程序状态`ListApplications`或者`DescribeApplication`行动。如果您的应用程序进入`RUNNING`status，但没有将数据写入到接收器，您可以通过添加Amazon CloudWatch 日志流到应用程序中。有关更多信息，请参阅 使用应用程序 CloudWatch 日志记录选项 (p. 207)。日志流包含可用于解决应用程序问题的消息。

快照、应用程序更新或应用程序停止错误：InvalidApplicationConfigurationException

在快照操作期间或创建快照的操作（例如更新或停止应用程序）期间，可能会出现类似下面的错误：

```
An error occurred (InvalidApplicationConfigurationException) when calling the
 UpdateApplication operation:

Failed to take snapshot for the application xxxx at this moment. The application is
 currently experiencing downtime.
Please check the application's CloudWatch metrics or CloudWatch logs for any possible
 errors and retry the request.
You can also retry the request after disabling the snapshots in the Kinesis Data Analytics
 console or by updating
```

```
the ApplicationSnapshotConfiguration through the Amazon SDK
```

在应用程序无法创建快照时，将会出现该错误。

如果在快照操作期间或创建快照的操作期间遇到该错误，请执行以下操作：

- 为应用程序禁用快照。您可以在 Kinesis Data Analytics 控制台中执行此操作，也可以使用SnapshotsEnabledUpdateparameterUpdateApplicationaction.
- 调查无法创建快照的原因。有关更多信息，请参阅 应用程序停滞在暂时状态 (p. 318)。
- 在应用程序恢复正常状态时，重新启用快照。

java.nio.file。NoSuchFileException: /usr/local/openjdk-8/lib/security/cacerts

在以前的部署中更新了 SSL 信任存储库位置。请在 ssl.truststore.location 参数中改用以下值：

```
/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
```

应用程序正在重启

如果你的应用程序不健康，那么它的 Apache Flink 作业会继续失败并重新启动。本节介绍了此情况的症状和故障排除步骤。

征兆

这种情况可能有以下症状：

- 这些区域有：FullRestarts指标不是零。此指标表示自您启动应用程序以来应用程序作业重新启动的次数。
- 这些区域有：Downtime指标不是零。此指标表示应用程序在FAILING要么RESTARTING状态。
- 应用程序日志包含的状态更改为RESTARTING要么FAILED. 您可以使用以下方法查询应用程序日志中的这些状态更改：CloudWatchLogs Insights 查询：分析错误：应用程序的任务相关故障 (p. 213).

原因和解决方案

以下情况可能会导致应用程序变得不稳定并反复重新启动：

- 操作员正在抛出异常：如果未处理应用程序中的操作符的任何异常，应用程序将进行故障转移（解释操作符无法处理该故障）。应用程序从最新的检查点重新启动，以保持“精确一次”的处理语义。因此，Downtime在这些重新启动期间不为零。为了防止发生这种情况，我们建议您在应用程序代码中处理任何可重试的异常。

  您可以查询应用程序日志以确定应用程序状态是否从 RUNNING 变为 FAILED，以调查发生这种情况的原因。有关更多信息，请参阅 the section called “分析错误：应用程序的任务相关故障” (p. 213)。
- Kinesis Data Streams 未正确配置：如果应用程序的源或接收器是 Kinesis 数据流，请检查指标对于直播ReadProvisionedThroughputExceeded要么WriteProvisionedThroughputExceeded错误消息。

  如果看到这些错误，可以通过增加 Kinesis 流的分片数量来增加该流的可用吞吐量。有关更多信息，请参阅 。如何更改 Kinesis Data Streams 中打开的分片的数量？.

- 其他来源或汇没有正确配置或不可用：检查是否正确预置应用程序预置源和接收器。检查是否使用应用程序中使用的任何源或接收器（例如其他Amazon服务或外部源或目标）预置得当，它们没有受到读取或写入限制，或者定期不可用。

  如果您遇到与依赖服务相关的吞吐量相关问题，请增加这些服务的可用资源，或调查任何错误或不可用的原因。
- 运营商没有正确配置：如果应用程序中某个操作员的线程上的工作负载未正确分配，则操作员可能会超载，应用程序可能会崩溃。有关调整运算符并行度的信息，请参阅正确管理运营商扩展 (p. 239).
- 应用程序失败DaemonException：如果您使用的是 1.11 之前的 Apache Flink 版本，则此错误会出现在应用程序日志中。您可能需要升级到 Apache Flink 的更高版本，以便使用 0.14 或更高版本的 KPL 版本。
- 应用程序失败TimeoutException、FlinkException，或者RemoteTransport异常：如果任务管理器崩溃，这些错误可能会出现在应用程序日志中。如果应用程序过载，任务管理器可能会遇到 CPU 或内存资源压力，从而导致他们失败。

  这些错误可能类似以下内容：

  - `java.util.concurrent.TimeoutException: The heartbeat of JobManager with id xxx timed out`
  - `org.apache.flink.util.FlinkException: The assigned slot xxx was removed`
  - `org.apache.flink.runtime.io.network.netty.exception.RemoteTransportException: Connection unexpectedly closed by remote task manager`

  要排查此情况，请检查以下内容：

  - 检查您的CloudWatchCPU 或内存使用量异常峰值的指标。
  - 检查应用程序的吞吐量问题。有关更多信息，请参阅 排查性能方 (p. 237)。
  - 检查应用程序日志是否存在应用程序代码引发的未处理的异常。
- 应用程序失败JaxbAnnotation找不到模块错误：如果您的应用程序使用 Apache Beam 但没有正确的依赖关系或依赖关系版本，则会出现此错误。使用 Apache Beam 的 Kinesis Data Analytics 应用程序必须使用以下版本的依赖项：

```
<jackson.version>2.10.2</jackson.version>
...
<dependency>
    <groupId>com.fasterxml.jackson.module</groupId>
    <artifactId>jackson-module-jaxb-annotations</artifactId>
    <version>2.10.2</version>
</dependency>
```

  如果您没有提供正确的版本`jackson-module-jaxb-annotations`作为显式依赖项，您的应用程序从环境依赖项加载它，由于版本不匹配，应用程序会在运行时崩溃。

  有关将 Apache Beam 与 Kinesis Data Analytics 一起使用的更多信息，请参阅使用CloudFormation使用 Kinesis Data Analytics (p. 113).

吞吐量太慢

如果您的应用程序处理传入流数据的速度不够快，它将表现不佳并变得不稳定。本节介绍此情况的症状和故障排除步骤。

征兆

这种情况可能为以下症状之一：

- 如果应用程序的数据源是 Kinesis 流，该流的`MillisBehindLatest`指标不断增加。

- 如果应用程序的数据源是 Amazon MSK 集群，则该集群的使用者滞后指标会持续增加。有关更多信息，请参阅 。使用器滞后监控中的Amazon MSK 开发人员指南.
- 如果应用程序的数据源是不同的服务或源，请检查任何可用的消费者滞后指标或可用数据。

原因和解决方案

应用程序吞吐量缓慢的原因可能有很多。如果应用程序无法跟上输入内容，请检查以下内容：

- 如果吞吐量滞后出现尖峰然后逐渐减弱，请检查应用程序是否正在重新启动。您的应用程序将在重新启动时停止处理输入，从而导致延迟峰值。有关应用程序故障的信息，请参阅应用程序正在重启 (p. 320).
- 如果吞吐量滞后一致，请检查您的应用程序是否针对性能进行了优化。有关优化应用程序性能的信息，请参阅排查性能方 (p. 237).
- 如果吞吐量滞后不是峰值而是持续增加，并且您的应用程序已针对性能进行了优化，则必须增加应用程序资源。有关增加应用程序资源的信息，请参阅扩缩 (p. 27).

有关应用程序源中吞吐量缓慢或使用者延迟增加的故障排除步骤，请参阅排查性能方 (p. 237).

无限制状态增长

如果您的应用程序没有正确处置过时的状态信息，它将持续积累，并导致应用程序性能或稳定性问题。本节介绍了此情况的症状和故障排除步骤。

征兆

这种情况可能出现以下症状：

- 这些区域有：`lastCheckpointDuration`指标正在逐渐增加或激增。
- 这些区域有：`lastCheckpointSize`指标正在逐渐增加或激增。

原因和解决方案

以下情况可能会导致您的应用程序累积州数据：

- 您的应用程序保留状态数据的时间超过了所需的时间。
- 您的应用程序使用持续时间太长的窗口查询。
- 你没有为州数据设置 TTL。有关更多信息，请参阅 。状态生存时间 (TTL)中的Apache Flink.
- 您运行的应用程序依赖于 Apache Beam 2.25.0 版本。您可以通过以下方式选择退出新版本的读转换扩展 Beam 应用程序属性有关键的实验和价值`use_deprecated_read`. 有关更多信息，请参阅 。Apache Beam.

有时，应用程序面临不断增长的状态规模增长，从长远来看是不可持续的（毕竟 Flink 应用程序无限期运行）。有时，这可以追溯到存储状态数据的应用程序，而没有正确老化旧信息。但是有时候对 Flink 可以提供的东西有不合理的期望。应用程序可以在跨越数天甚至几周的大时间窗口内使用聚合。除非AggregregatFuncate用于允许增量聚合，Flink 需要使整个窗口的事件保持状态。

此外，在使用流程函数实现自定义运算符时，应用程序需要从业务逻辑不再需要的状态中删除数据。在那种情况下，状态生存时间可用于根据处理时间自动删除数据的老化。Kinesis Data Analytics 正在使用增量检查点，因此状态 ttl 基于RockSDB 压缩. 只有在进行压缩操作后，您才能观察到状态大小的实际减少（以检查点大小表示）。特别是对于 200 MB 以下的检查点，由于状态即将到期，您不可能观察到检查点大小缩小的情况。但是，保存点基于不包含旧数据的状态的干净副本，因此您可以在 Kinesis Data Analytics 中触发快照以强制删除过时的状态。

出于调试目的，禁用增量检查点以更快地验证检查点大小实际上减小还是稳定（并避免 RockSB 中压缩的影响）是有意义的。但是，这需要向服务团队提供票证。

I/O 绑定运算符

最好避免在数据路径上依赖外部系统。保持参考数据集处于状态，而不是查询外部系统来丰富个别事件，通常性能要高得多。但是，有时候，有些依赖项无法轻松移动到状态，例如，如果您想使用 Amazon Sageemaker 上托管的机器学习模型丰富事件。

通过网络与外部系统接口的运营商可能会成为瓶颈并造成背压。强烈建议使用AsyncIO以实现该功能，以减少单个呼叫的等待时间并避免整个应用程序放慢速度。

此外，对于具有 I/O 绑定运算符的应用程序，增加ParallelismPerKPUKinesis Data Analytics 应用程序的设置。此配置描述应用程序在其每个 Kinesis 处理单元 (KPU) 可以执行的 parallel 行子任务数。通过将值从默认值 1 增加到（比如 4），应用程序利用了相同的资源（并且具有相同的成本），但可以扩展到并行度的 4 倍。这对于 I/O 绑定的应用程序来说很有效，但它会给未绑定 I/O 的应用程序带来额外的开销。

来自 Kinesis 数据流的上游或源限制

症状：应用程序正在遇到`LimitExceededExceptions`来自他们的上游源 Kinesis 数据流。

潜在原因：Apache Flink 库 Kinesis 连接器的默认设置被设置为从 Kinesis 数据流源读取，默认设置非常激进，每个获取的最大记录数`GetRecords`调用。默认情况下，Apache Flink 配置为每个获取 10,000 条记录 GetRecords 调用（默认情况下此调用每 200 毫秒一次），尽管每个分片的限制只有 1,000 条记录。

当尝试从 Kinesis 数据流中消耗时，这种默认行为可能会导致限制，这将影响应用程序的性能和稳定性。

这可以通过检查 CloudWatch `ReadProvisionedThroughputExceeded`指标，并查看此指标大于零的长期或持续时间。

客户还可以在中看到这个 CloudWatch 通过看到继续了解他们的 Kinesis Analytics Flink 应用程序`LimitExceededException`错误消息。

解决方法：客户可以执行以下两项操作之一来解决此情况：

- 降低每个获取的记录数的默认限制 GetRecords 呼叫
- 客户可以在其 Kinesis Analytics Flink 应用程序中启用自适应读取。有关自适应读取功能的更多信息，请参阅SHARD_USE_ADAPTIVE_READ

检查点

检查点是 Flink 用来确保应用程序状态容错的机制。该机制允许 Flink 在作业失败时恢复操作员状态，并为应用程序提供与无故障执行相同的语义。借助 Kinesis Data Analytics，应用程序的状态存储在 RocksDB 中，RocksDB 是一种嵌入式键/值存储，可将其工作状态保留在磁盘上。使用检查点时，状态也会上传到 Amazon S3，因此即使磁盘丢失，也可以使用检查点来恢复应用程序状态。

有关更多信息，请参阅 。状态快照如何工作？.

检查点阶段

对于 Flink 中的检查点操作员子任务，主要有 5 个阶段：

- 等待 [启动延迟] — Flink 使用插入到流中的检查点屏障，因此此阶段的时间是操作员等待检查点屏障到达的时间。

- 对齐 [对齐Durat] — 在这个阶段，子任务已经达到了一个障碍，但它正在等待来自其他输入流的屏障。
- 同步检查点 [Sync] — 此阶段是指子任务实际拍摄操作员状态的快照，并阻止子任务上的所有其他活动。
- Async 检查点 [异步Duration] — 此阶段的大部分时间是将状态上传到 Amazon S3 的子任务。在此阶段，子任务不再被阻止，可以处理记录。
- 确认 — 这通常是一个短暂的阶段，只是子任务向 JobManager 并执行任何提交消息（例如使用Kafka接收器）。

每个阶段（确认除外）都映射到 Flink WebUI 提供的检查点持续时间度量，这有助于隔离检查点长的原因。

要查看检查点上每个可用指标的精确定义，请转到历史选项卡.

正在调查

在调查检查点持续时间较长时，要确定的最重要的事情是检查点的瓶颈，即哪个操作员和子任务在检查点上花费的时间最长，以及该子任务的哪个阶段需要较长的时间。这可以通过作业检查点任务下的 Flink WebUI 来确定。Flink 的 Web 界面提供了有助于调查检查点问题的数据和信息。有关完整的细分，请参阅监控检查点.

首先要看的是端到端持续时间Job 图中的每个操作员，以确定哪个操作员需要很长时间才能进行检查并值得进一步调查。根据 Flink 文档，持续时间的定义是：

从触发时间戳到最后一次确认的持续时间（如果尚未收到确认，则为 n/a）。完整检查点的端到端持续时间由确认该检查点的最后一个子任务决定。此时间通常大于单个子任务实际检查点状态所需的时间。

检查点的其他持续时间也提供了有关时间花在何处的更精细的信息。

如果Sync为高，则表示快照期间发生了什么事。在此阶段`snapshotState()`为实现 SnapshotState 接口的类调用；这可以是用户代码，因此线程转储可用于调查此问题。

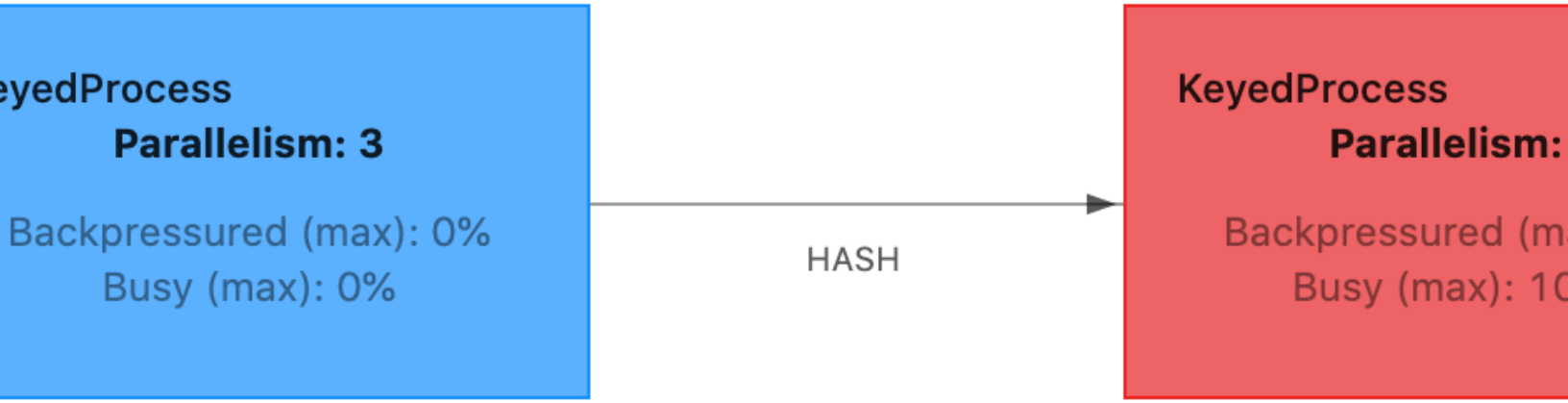
长整型异步Duration这表明在将状态上传到 Amazon S3 上花费了大量时间。如果状态很大，或者有大量的状态文件正在上传，则可能会出现这种情况。如果是这样的话，那么值得研究一下应用程序是如何使用状态的，并确保在可能的情况下使用 Flink 原生数据结构（使用键控状态)。Kinesis Data Analytics 配置 Flink 的方式可以最大限度地减少 Amazon S3 调用的次数，以确保调用时间不会太长。下面是一个操作员检查点统计信息的示例。它表明异步Duration与前面的操作员检查点统计数据相比相对较长。

ata Size	Sync Duration	Async Duration	Processed (p
	8ms	357ms	0 B (0 B)
	28ms	653ms	0 B (0 B)
	69ms	1s	0 B (0 B)

ted Data	Sync Duration	Async Duration	Processed (persisted) Data
	8ms	429ms	0 B (0 B)
	69ms	1s	0 B (0 B)
	8ms	357ms	0 B (0 B)

	1/1 (100%)	2022-03-02 14:16:49	131ms

这些区域有：启动延迟较高将表明大部分时间都花在等待检查站屏障到达操作员身上。这表明应用程序需要一段时间来处理记录，这意味着屏障正在缓慢地流过作业图。如果Job 压力过大或操作员经常忙碌，通常会出现这种情况。下面是一个示例 JobGraph second KeyedProcess 运算符正忙。


eyedProcess
Parallelism: 3
Backpressured (max): 0%
Busy (max): 0%
HASH
KeyedProcess
Parallelism:
Backpressured (m
Busy (max): 10

你可以使用 Flink Flame Graphs 或 TaskManager 线程转储。一旦确定了瓶颈，就可以使用 Flame-Graphs 或线程转储对其进行进一步调查。

线程转储

线程转储是另一种调试工具，其级别略低于火焰图。线程转储输出所有线程在某个时间点的执行状态。Flink 接受一个 JVM 线程转储，这是 Flink 进程中所有线程的执行状态。线程的状态由线程的堆栈跟踪以及一些其他信息表示。火焰图实际上是使用快速连续拍摄的多个堆栈轨迹构建的。该图是由这些轨迹构成的可视化项，可以轻松识别常见的代码路径。

```
"KeyedProcess (1/3)#0" prio=5 Id=1423 RUNNABLE
    at app//scala.collection.immutable.Range.foreach$mVc$sp(Range.scala:154)
    at $line33.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:19)
    at $line33.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:14)
    at app//
org.apache.flink.streaming.api.operators.KeyedProcessOperator.processElement(KeyedProcessOperator.java:
    at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStreamTaskN
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTaskNetwork
    at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProcessor.java
    ...
```

上面是从 Flink UI 中获取的单个线程的线程转储片段。第一行包含有关此线程的一些常规信息，包括：

- ThreadKeyedProcess (1/3) #0
- 线程的优先级prio=5
- 一个唯一的话题编号Id=1423
- 线程状态可运行

线程的名称通常提供有关线程一般用途的信息。操作员线程可以通过其名称来标识，因为操作员线程与运算符具有相同的名称，以及它与哪个子任务相关的指示，例如KeyedProcess (1/3) #0线程来自KeyedProcess运算符和来自第 1 个（共 3 个）子任务。

线程可以处于以下某种状态：

- NEW — 线程已创建但尚未处理
- 可运行 — 线程在 CPU 上执行
- BLOCKED — 线程正在等待另一个线程释放其锁定
- WAITING — 线程正在等待`wait()`、`join()`，或者`park()`方法
- TIMED_WAITING — 线程正在使用休眠、等待、加入或暂留方法等待，但等待时间最长。

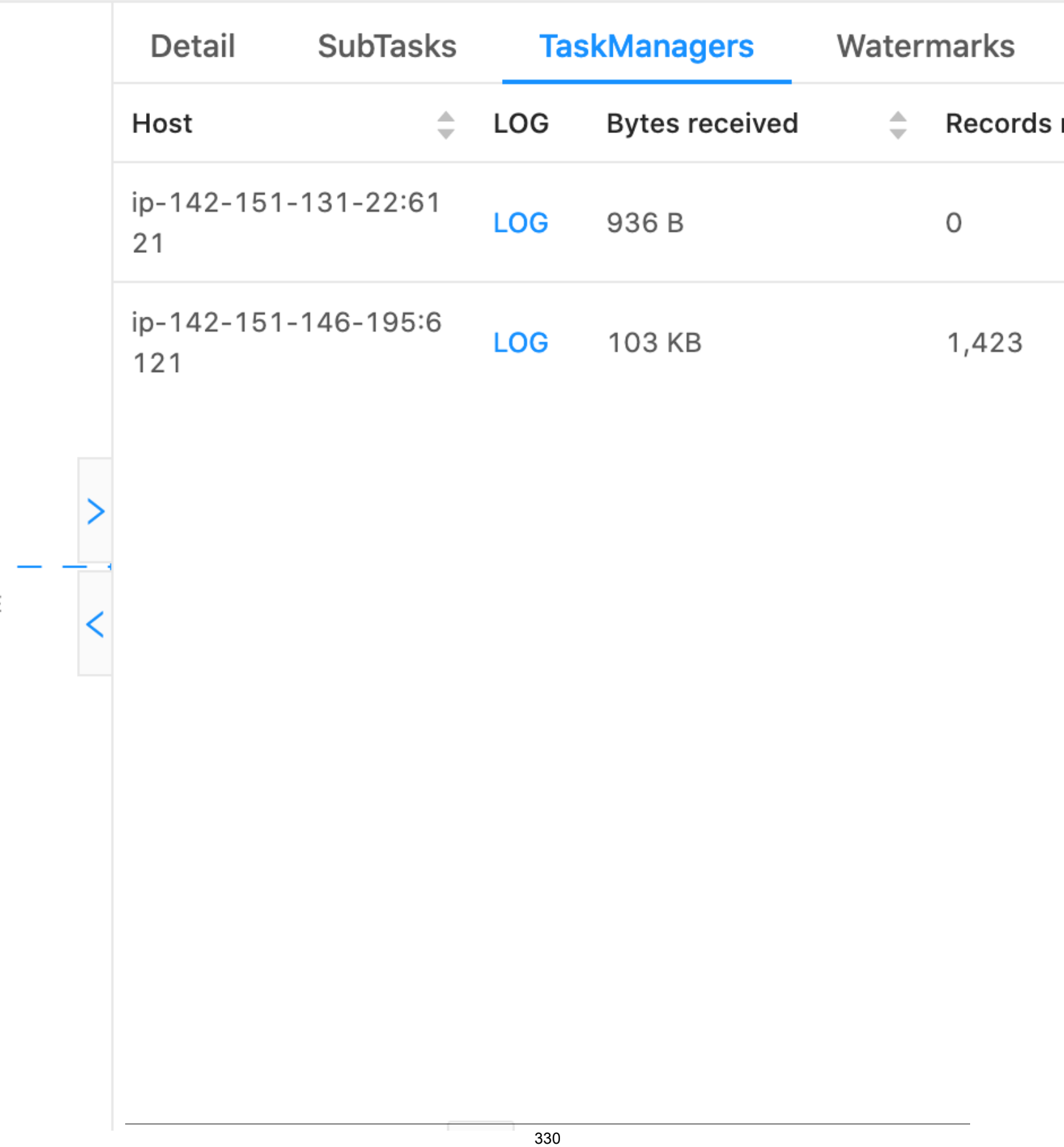
> **Note**
>
> 在 Flink 1.13 中，线程转储中单个堆栈跟踪的最大深度限制为 8。

> **Note**
>
> 线程转储应该是调试 Flink 应用程序中性能问题的最后手段，因为线程转储很难阅读，需要采集多个样本并进行手动分析。如果可能的话，最好使用火焰图。

Flink 中的线程转储

在 Flink 中，可以通过选择任务管理器选项，在 Flink UI 的左侧导航栏上选择一个特定的任务管理器，然后导航到线程转储选项卡。线程转储可以下载，复制到你最喜欢的文本编辑器（或线程转储分析器），或者直接在 Flink Web UI 的文本视图中进行分析（但是，最后一个选项可能有点笨拙。

要确定使用哪个任务管理器进行线程转储TaskManagers选项卡可以在选择特定运算符时使用。这表明操作员在操作员的不同子任务上运行，并且可以在不同的任务管理器上运行。


Detail
SubTasks
TaskManagers
Watermarks
Host
LOG
Bytes received
Records
ip-142-151-131-22:6121
LOG
936 B
0
ip-142-151-146-195:6121
LOG
103 KB
1,423

转储将由多个堆栈跟踪组成。但是，在调查垃圾场时，与操作员有关的问题是最重要的。这些很容易找到，因为操作员线程与运算符具有相同的名称，并且指示它与哪个子任务相关。例如，以下堆栈跟踪来自KeyedProcess运算符和是第一个子任务。

```
"KeyedProcess (1/3)#0" prio=5 Id=595 RUNNABLE
    at app//scala.collection.immutable.Range.foreach$mVc$sp(Range.scala:155)
    at $line360.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:19)
    at $line360.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:14)
    at app//
org.apache.flink.streaming.api.operators.KeyedProcessOperator.processElement(KeyedProcessOperator.java:
    at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStreamTaskN
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTaskNetwork
    at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProcessor.java
    ...
```

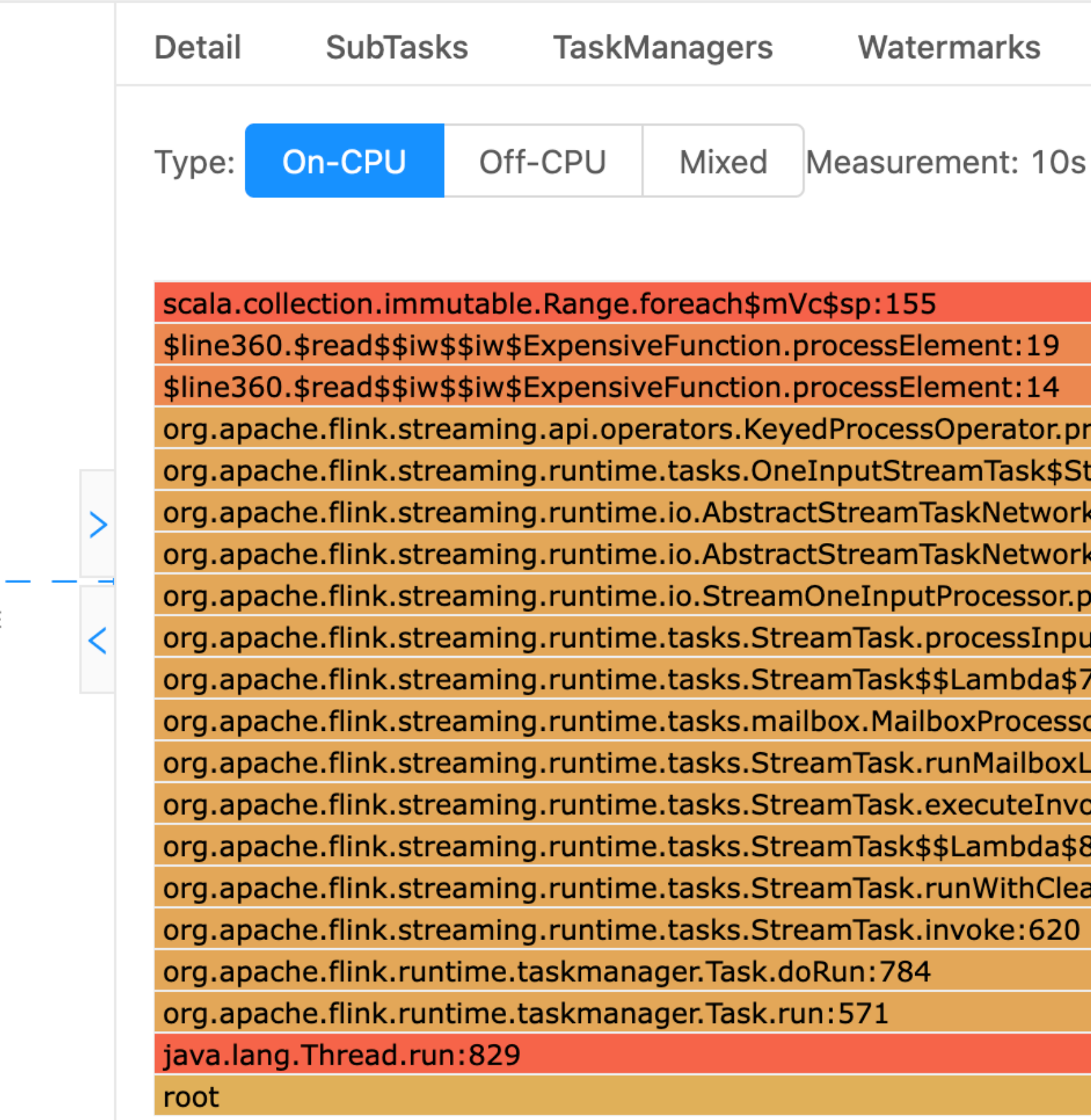
如果有多个具有相同名称的运算符，这可能会变得混乱，但是我们可以命名运算符来解决这个问题。例如：

```
....
.process(new ExpensiveFunction).name("Expensive function")
```

火焰图

火焰图是一种有用的调试工具，它可视化目标代码的堆栈跟踪，从而可以识别最频繁的代码路径。它们是通过对堆栈轨迹进行多次采样而创建的。火焰图的x轴显示不同的堆栈轮廓，而y轴显示堆叠深度和堆栈追踪中的调用。火焰图中的单个矩形表示堆叠框架，框架的宽度显示它在堆栈中出现的频率。有关火焰图及其使用方法的更多信息，请参阅火焰图.

在 Flink 中，可以通过 Web UI 访问操作员的火焰图，方法是选择一个操作员，然后选择FlameGraph选项卡。一旦采集了足够的样本，火焰图就会显示出来。以下为 FlameGraph (对于) ProcessFunction 这花了很多时间去检查站。


Detail
SubTasks
TaskManagers
Watermarks
Type:
On-CPU
Off-CPU
Mixed
Measurement: 10s
scala.collection.immutable.Range.foreachmVcsp:155
$line360.$read$$iw$$iw$ExpensiveFunction.processElement:19
$line360.$read$$iw$$iw$ExpensiveFunction.processElement:14
org.apache.flink.streaming.api.operators.KeyedProcessOperator.pr
org.apache.flink.streaming.runtime.tasks.OneInputStreamTask$St
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetwork
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetwork
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.p
org.apache.flink.streaming.runtime.tasks.StreamTask.processInpu
org.apache.flink.streaming.runtime.tasks.StreamTask$$Lambda$7
org.apache.flink.streaming.runtime.tasks.mailbox.MailboxProcesso
org.apache.flink.streaming.runtime.tasks.StreamTask.runMailboxL
org.apache.flink.streaming.runtime.tasks.StreamTask.executeInvo
org.apache.flink.streaming.runtime.tasks.StreamTask$$Lambda$8
org.apache.flink.streaming.runtime.tasks.StreamTask.runWithClea
org.apache.flink.streaming.runtime.tasks.StreamTask.invoke:620
org.apache.flink.runtime.taskmanager.Task.doRun:784
org.apache.flink.runtime.taskmanager.Task.run:571
java.lang.Thread.run:829
root

这是一个非常简单的火焰图，它显示所有的 CPU 时间都花在 foreach 外观中`processElement`的 ExpensiveFunction 运算符。您还可以获得行号，以帮助确定代码执行的执行位置。

Checkpointing 已超时

如果您的应用程序未经过优化或未正确配置，则检查点可能会失败。本节介绍此情况的症状和故障排除步骤。

征兆

如果应用程序的检查点失败，`numberOfFailedCheckpoints`将大于零。

检查点失败的原因可能是直接故障（例如应用程序错误），也可能是由于暂时性故障（例如应用程序资源耗尽）所致。检查应用程序日志和指标以了解以下症状：

- 您的代码中存在错误。
- 访问应用程序的依赖服务时出错。
- 序列化数据时出错。如果默认序列化器无法序列化应用程序数据，则应用程序将失败。有关在应用程序中使用自定义序列化器的信息，请参阅自定义序列化器中的Apache Flink 文档.
- 内存不足错误。
- 以下指标出现峰值或稳步增长：
  - `heapMemoryUtilization`
  - `oldGenerationGCTime`
  - `oldGenerationGCCount`
  - `lastCheckpointSize`
  - `lastCheckpointDuration`

有关监控检查点的更多信息，请参阅监控检查点中的Apache Flink 文档.

原因和解决方案

您的应用程序日志错误消息显示了直接失败的原因。暂时性故障可能有以下原因：

- 您的应用程序的 KPU 配置不足。有关增加应用程序配置的信息，请参阅扩缩 (p. 27).
- 应用程序状态大小过大。您可以使用监控应用程序状态大小`lastCheckpointSize`指标。
- 应用程序的状态数据在密钥之间分布不均。如果您的应用程序使用`KeyBy`操作员，请确保您的传入数据在密钥之间平均分配。如果将大部分数据分配给单个密钥，则会产生瓶颈，从而导致失败。
- 您的应用程序遇到内存或垃圾回收反压。监视你的应用程序的`heapMemoryUtilization`、`oldGenerationGCTime`, 和`oldGenerationGCCount`用于峰值或稳步增加的值。

Apache Beam 应用程序的检查点失败

如果您的 Beam 应用程序配置了shutdownSourcesAfterIdleMs设置为 0ms，则检查点可能无法触发，因为任务处于“已完成”状态。本节介绍这种情况的症状和解决方法。

症状

转到 Kinesis Data Analytics 应用程序 CloudWatch 记录并检查是否记录了以下日志消息。以下日志消息表明，由于某些任务已完成，检查点无法触发。

```
                {
                "locationInformation":
"org.apache.flink.runtime.checkpoint.CheckpointCoordinator.onTriggerFailure(CheckpointCoordinator.java
                "logger": "org.apache.flink.runtime.checkpoint.CheckpointCoordinator",
                "message": "Failed to trigger checkpoint for job your job ID since some
tasks of job your job ID has been finished, abort the checkpoint Failure reason: Not all
required tasks are currently running.",
                "threadName": "Checkpoint Timer",
                "applicationARN": your application ARN,
                "applicationVersionId": "5",
                "messageSchemaVersion": "1",
                "messageType": "INFO"
                }
```

这也可以在 Flink 仪表板上找到，其中一些任务已进入 “已完成” 状态，现在无法进行检查点操作。

TaskManagers | Watermarks | Accumulators | BackPressure | Metrics | FlameGraph

ived	Records Received	Bytes Sent	Records Sent	Attempt	Host	Start Time	Dura
	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	13m
	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	3s
	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	3s
	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	3s
	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	3s

原因

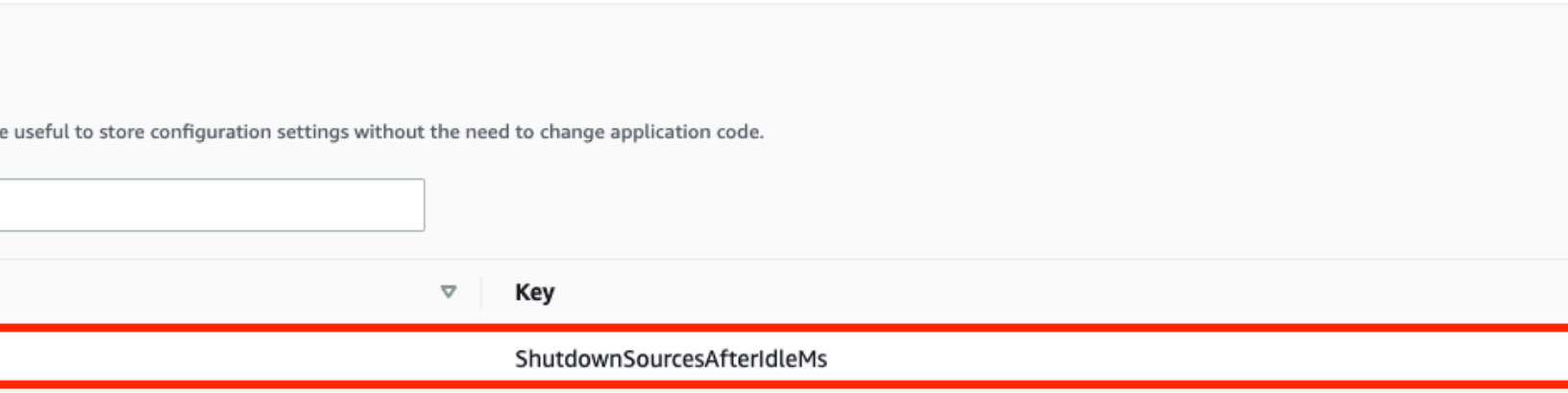
shutdownSourcesAfterIdleMs 是一个 Beam 配置变量，用于关闭已在配置的毫秒时间内处于空闲状态的源。一旦源被关闭，便无法再进行检查点操作。这可能导致检查点失败.

任务进入 “已完成” 状态的原因之一是 shutdownSourcesAfterIdleMs 设置为 0ms，这表示空闲的任务将立即关闭。

解决方案

要防止任务立即进入 “已完成” 状态，请将 shutdownSourcesAfterIdleMs 改为 long.max_VALUE。可通过两种方式执行此任务：

- 选项 1：如果在 Kinesis Data Analytics 应用程序配置页面中设置了光束配置，则可以添加新的键值对进行设置 shutdpwnSourcesAfteridleMs 如下所示：



- 选项 2：如果在 JAR 文件中设置了光束配置，则可以设置 shutdownSourcesAfterIdleMs 如下所示：

```
                        FlinkPipelineOptions options =
PipelineOptionsFactory.create().as(FlinkPipelineOptions.class); // Initialize Beam
Options object

                        options.setShutdownSourcesAfterIdleMs(Long.MAX_VALUE); // set
shutdownSourcesAfterIdleMs to Long.MAX_VALUE
                        options.setRunner(FlinkRunner.class);

                        Pipeline p = Pipeline.create(options); // attach specified
options to Beam pipeline
```

背压

Flink 使用背压来调整单个操作员的处理速度。

出于多种原因，操作员可能难以继续处理收到的消息量。该操作可能需要比操作员可用的 CPU 资源多，操作员可能会等待 I/O 操作完成。如果操作员无法以足够快的速度处理事件，则会在上游操作员中建立反压，并向慢速运算符提供反压。这会导致上游运算符减速，从而进一步将反压传播到源，并通过减慢速度使源适应应用程序的整体吞吐量。你可以找到关于背压及其工作原理的更深入的描述Apache Flink™ 如何处理背压.

了解应用程序中的哪些运算符运行缓慢，可以为您提供重要信息，以了解应用程序中性能问题的根本原因。背压信息是通过 Flink 控制面板公开. 要识别慢速运算符，请查找具有最接近汇的高背压值的运算符（以下示例中的运算符 B）。因此，造成缓慢的运算符是下游运算符之一（示例中的运算符 C）。B 可以更快地处理事件，但由于无法将输出转发给实际的慢速运算符 C 而受到反压。

```
A (backpressured 93%) -> B (backpressured 85%) -> C (backpressured 11%) -> D (backpressured
 0%)
```

一旦你确定了慢速操作员，试着去理解为什么它很慢。可能有各种各样的原因，有时不清楚出了什么问题，可能需要数天的调试和分析才能解决。以下是一些显而易见且更常见的原因，其中一些将在下面进一步解释：

- 操作员正在执行缓慢的 I/O，例如网络调用（考虑改用 AsynCIO）。
- 数据存在偏差，一个操作员收到的事件比其他操作员多（通过在 Flink 控制面板中查看单个子任务（即同一个运算符的实例）的进出消息数量进行验证。
- 这是一项资源密集型操作（如果没有数据偏差，可以考虑向外扩展 CPU/内存限制工作或增加`ParallelismPerKPU`用于 I/O 绑定工作）

- 操作员的广泛日志记录（将生产应用程序的日志记录减少到最低限度，或者考虑将调试输出发送到数据流）。

使用丢弃接收器测试吞吐量

这些区域有：丢弃接收器只是在执行应用程序时忽略它收到的所有事件（没有任何接收器的应用程序无法执行）。这对于吞吐量测试、分析以及验证应用程序是否正常扩展非常有用。这也是一种非常务实的健全性检查，用于验证水槽是否造成背压或应用（但仅仅检查背压指标通常更容易、更直接）。

通过将应用程序的所有接收器替换为丢弃的接收器，并创建生成类似于生产数据的数据的模拟源，您可以测量特定并行度设置下应用程序的最大吞吐量。然后，您还可以增加并行度，以验证应用程序是否可以正确扩展，并且没有仅在吞吐量较高时才会出现的瓶颈（例如，由于数据偏差）。

数据偏斜

Flink 应用程序以分布式方式在集群上执行。为了向外扩展到多个节点，Flink 使用了密钥流的概念，这意味着流的事件根据特定的键（例如客户 ID）进行分区，然后 Flink 可以在不同的节点上处理不同的分区。然后，许多 Flink 运算符会根据这些分区进行评估，例如带钥匙的窗户、流程函数和异步I/O.

选择分区键通常取决于业务逻辑。同时，还有许多最佳实践，例如DynamoDB和 Spark，同样适用于 Flink，包括：

- 确保分区键的基数较高
- 避免分区之间的事件卷出现偏差

您可以通过在 Flink 控制面板中比较子任务（即同一个运算符的实例）的接收/发送记录来识别分区中的偏差。此外，可以将 Kinesis Data Analytics 监控配置为公开`numRecordsIn/Out`和`numRecordsInPerSecond/OutPerSecond`在子任务级别也是如此。

状态偏差

对于有状态运算符，即为其业务逻辑（如窗口）维护状态的运算符，数据偏斜总是会导致状态偏差。由于数据偏斜，某些子任务比其他子任务接收更多的事件，因此也将更多的数据保留在状态中。但是，即使对于分区均匀平衡的应用程序，在状态下保留的数据量也可能存在偏差。例如，对于会话窗口，某些用户和会话可能分别比其他用户和会话长得多。如果较长的会话恰好属于同一个分区，则可能导致同一运算符的不同子任务保持的状态大小不平衡。

状态偏差不仅会增加单个子任务所需的更多内存和磁盘资源，还会降低应用程序的整体性能。当应用程序使用检查点或保存点时，操作员状态将保留到 Amazon S3 中，以保护状态免受节点或集群故障的影响。在此过程中（尤其是在 Kinesis Data Analytics 上默认启用一次语义的情况下），从外部角度来看，处理将暂停，直到检查点/保存点完成。如果存在数据偏差，则完成操作的时间可能受单个子任务的限制，该子任务已累积了特别多的状态。在极端情况下，获取检查点/保存点可能会因为单个子任务无法保持状态而失败。

与数据偏斜类似，状态偏斜会大大减慢应用程序的速度。

要识别状态偏差，您可以利用 Flink 控制面板。查找最近的检查点或保存点，并在详细信息中比较为各个子任务存储的数据量。

JAR 超过 512 MB

有时候，您可能需要将在 Kinesis Analytics 上运行的 Flink 应用程序打包到多个 JAR 文件中，而不是单个 JAR 文件中。如果以下是我们通过客户上报注意到的这些用例的几个示例：512MB 硬限制会限制您在 Kinesis Analytics 上运行应用程序的能力，请考虑以下解决方法：

总会有一个 Flink 客户端在运行。它采用 Flink 应用程序的代码，将其转换为 JobGraph 并将其提交给 JobManager. 这些区域有： JobManager 将作品分发到 TaskManagers，其中运行实际的运算符（例如源、变换和汇点）。

你可以在 Flink 客户端上运行 Java 代码，它会下载其他 JAR 并通过创建另一个 JAR 将它们添加到类路径中 ClassLoader，或者将其保存到 Flink Cache blob 存储区并将其传递给 TaskManagers 每个任务管理器都可以在其中创建这些对象的实例`open()`的方法 ProcessFunction.

有关示例，请参阅 。在运行时添加下载 flink jar.

整合不同区域中的资源

您可以使用`StreamingFileSink`通过 Flink 配置中跨区域复制所需的设置，写入与 Kinesis Data Analytics 应用程序位于不同区域的 Amazon S3 存储桶。要执行此操作，请在以下位置提交支持票证Amazon支持中心.

Apache Flink 的 Amazon Kinesis Data Analytics 文档历史记录

下表列出了自 Amazon Kinesis Data Analytics 上一次发布以来对文档所做的重要更改。

- API 版本：2018-05-23
- 最近文档更新时间：2021 年 10 月 13 日

更改	描述	日期
Support Apache Flink 版本 1.13.2	Kinesis Data Analytics 现在支持使用 Apache Flink 版本 1.13.2 的应用程序。使用 Apache Flink 表 API 创建 Kinesis Data Analytics 应用程序。有关更多信息，请参阅 创建应用程序 (p. 3)。	2021 年 10 月 13 日
Support Python	Kinesis Data Analytics 现在支持将 Python 与 Apache Flink 表 API 和 SQL 结合使用的应用程序。有关更多信息，请参阅 使用 Python (p. 15)。	2021 年 3 月 25 日
Support Apache Flink 1.11.1	现在 Kinesis Data Analytics 支持使用 Apache Flink 1.11.1 的应用程序。使用 Apache Flink 表 API 创建 Kinesis Data Analytics 应用程序。有关更多信息，请参阅 创建应用程序 (p. 3)。	2020 年 11 月 19 日
Apache Flink 仪表板	使用 Apache Flink 控制面板监控应用程序的运行状况和性能。有关更多信息，请参阅 Apache Flink 控制面板 (p. 38)。	2020 年 11 月 19 日
使用者 EFO	创建使用增强型扇出 (EFO) 消费者从 Kinesis 数据流中读取的应用程序。有关更多信息，请参阅 EFO cumer (p. 149)。	2020 年 10 月 6 日
Apache Beam	创建使用 Apache Beam 处理流数据的应用程序。有关更多信息，请参阅 使用CloudFormation使用 Kinesis Data Analytics (p. 113)。	2020 年 9 月 15 日
性能	如何解决应用程序性能问题，以及如何创建高性能应用程序。有关更多信息，请参阅 性能 (p. 237)。	2020 年 7 月 21 日
自定义密钥库	如何访问在传输过程中使用自定义密钥库进行加密的 Amazon	2020 年 6 月 10 日

更改	描述	日期
	MSK 集群。有关更多信息，请参阅 自定义信任库 (p. 171)。	
CloudWatch 警报	关于创建的建议CloudWatch使用针对 Apache Flink 的 Kinesis Data Analytics 的警报。有关更多信息，请参阅 告警 (p. 226)。	2020 年 6 月 5 日
新 CloudWatch 指标	Kinesis Data Analytics 现在向亚马逊发布 22 项指标 CloudWatchMetrics (指标)。有关更多信息，请参阅 指标与维度 (p. 214)。	2020 年 5 月 12 日
自定义 CloudWatch 指标	定义特定于应用程序的指标并将其发送给亚马逊 CloudWatchMetrics (指标)。有关更多信息，请参阅 自定义 指标 (p. 223)。	2020 年 5 月 12 日
例如：在不同账户中从 Kinesis 流中读取	了解如何在不同版本中访问 Kinesis 流Amazon帐户在 Kinesis Data Analytics 应用程序中。有关更多信息，请参阅 跨账户 (p. 166)。	2020 年 3 月 30 日
支持 Apache Flink 1.8.2	现在 Kinesis Data Analytics 支持使用 Apache Flink 1.8.2 的应用程序。使用 FlinkStreamingFile将输出直接写入到 S3 中的接收器。有关更多信息，请参阅 创建应用程序 (p. 3)。	2019 年 12 月 17 日
VPC Kinesis Data Analytics	配置 Kinesis Data Analytics 应用程序以连接到 Virtual Private Cloud。有关更多信息，请参阅 使用 Amazon VPC (p. 311)。	2019 年 11 月 25 日
Kinesis Data Analytics 最佳实践	创建和管理 Kinesis Data Analytics 应用程序的最佳实践。有关更多信息，请参阅 最佳实践 (p. 246)。	2019 年 10 月 14 日
应用程序 Kinesis Data Analytics 应用程序分析	使用CloudWatch记录见解以监控 Kinesis Data Analytics 应用程序。有关更多信息，请参阅 分析日志 (p. 211)。	2019 年 6 月 12 日
应用程序 Kinesis Data Analytics 应用	在 Apache Flink 的 Kinesis Data Analytics 中使用运行时属性。有关更多信息，请参阅 运行时属性 (p. 18)。	2019 年 6 月 24 日
标记 Kinesis Data Analytics 应用程序	使用应用程序标记来确定每个应用程序的成本，控制访问，或用于用户定义的目的。有关更多信息，请参阅 使用标记 (p. 30)。	2019 年 5 月 8 日

更改	描述	日期
应用 Kinesis Data Analytics 示例	示例应用程序，示例应用程序，说明窗口操作符并将输出写入到CloudWatch日志。有关更多信息，请参阅 示例 (p. 123)。	2019 年 5 月 1 日
使用记录 Kinesis Data Analytics API 调用Amazon CloudTrail	Amazon Kinesis Data Analytics与Amazon CloudTrail，提供用户、角色或用户所执行操作的记录的服务Amazon服务在 Kinesis Data Analytics 中。有关更多信息，请参阅 使用 Amazon CloudTrail (p. 234)。	2019 年 3 月 22 日
创建应用程序（Kinesis Data Firehose 接收器）	练习创建 Apache Flink 的 Kinesis Data Analytics 应用程序，其中将 Amazon Kinesis 数据流作为源，并将 Amazon Kinesis Data Firehose 传输流作为汇。有关更多信息，请参阅 Kinesis Data Firehose (p. 155)。	2018 年 12 月 13 日
公开发行版	这是初始版本Kinesis Data Analytics Java 应用程序开发人员指南.	2018 年 11 月 27 日

Kinesis Data Analytics API 示例代码

本主题包含 Kinesis Data Analytics 操作的示例请求块。

要在 Amazon Command Line Interface (Amazon CLI) 中将 JSON 作为一个操作的输入，请将请求保存在 JSON 文件中。然后，使用 `--cli-input-json` 参数将文件名传递给该操作。

以下示例说明了如何将 JSON 文件与操作一起使用。

```
$ aws kinesisanalyticsv2 start-application --cli-input-json file://start.json
```

有关使用 JSON 的更多信息，请参阅Amazon CLI，请参阅生成 CLI 框架和 CLI 输入 JSON 参数中的Amazon Command Line Interface用户指南.

主题

AddApplicationCloudWatchLoggingOption

请求代码的以下示例请求代码：AddApplicationCloudWatchLoggingOption行动添加了亚马逊 CloudWatchKinesis Data Analytics 应用程序的日志记录选项：

```
{
```

```
    "ApplicationName": "MyApplication",
    "CloudWatchLoggingOption": {
        "LogStreamARN": "arn:aws:logs:us-east-1:123456789123:log-group:my-log-group:log-
stream:My-LogStream"
    },
    "CurrentApplicationVersionId": 2
}
```

AddApplication输入

请求代码的以下示例请求代码：AddApplication输入操作将应用程序输入添加到 Kinesis Data Analytics 应用程序中：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "Input": {
      "InputParallelism": {
         "Count": 2
      },
      "InputSchema": {
         "RecordColumns": [
            {
               "Mapping": "$.TICKER",
               "Name": "TICKER_SYMBOL",
               "SqlType": "VARCHAR(50)"
            },
            {
                "SqlType": "REAL",
                "Name": "PRICE",
                "Mapping": "$.PRICE"
            }
         ],
         "RecordEncoding": "UTF-8",
         "RecordFormat": {
            "MappingParameters": {
               "JSONMappingParameters": {
                  "RecordRowPath": "$"
               }
            },
            "RecordFormatType": "JSON"
         }
      },
      "KinesisStreamsInput": {
         "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/ExampleInputStream"
      }
   }
}
```

AddApplicationInputProcessing配置

请求代码的以下示例请求代码：AddApplicationInputProcessing配置操作将应用程序输入处理配置添加到 Kinesis Data Analytics 应用程序：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
```

```
   "InputId": "2.1",
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "arn:aws:lambda:us-east-1:012345678901:function:MyLambdaFunction"
      }
   }
}
```

AddApplication输出

请求代码的以下示例请求代码：AddApplication输出操作将 Kinesis 数据流作为应用程序输出添加到 Kinesis Data Analytics 应用程序中：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "Output": {
      "DestinationSchema": {
         "RecordFormatType": "JSON"
      },
      "KinesisStreamsOutput": {
         "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/ExampleOutputStream"
      },
      "Name": "DESTINATION_SQL_STREAM"
   }
}
```

AddApplicationReferenceData源

请求代码的以下示例请求代码：AddApplicationReferenceData源操作将 CSV 应用程序引用数据源添加到 Kinesis Data Analytics 应用程序中：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5,
   "ReferenceDataSource": {
      "ReferenceSchema": {
         "RecordColumns": [
            {
               "Mapping": "$.TICKER",
               "Name": "TICKER",
               "SqlType": "VARCHAR(4)"
            },
            {
               "Mapping": "$.COMPANYNAME",
               "Name": "COMPANY_NAME",
               "SqlType": "VARCHAR(40)"
            },
         ],
         "RecordEncoding": "UTF-8",
         "RecordFormat": {
            "MappingParameters": {
               "CSVMappingParameters": {
                  "RecordColumnDelimiter": " ",
                  "RecordRowDelimiter": "\r\n"
               }
            },
```

```
            "RecordFormatType": "CSV"
         }
      },
      "S3ReferenceDataSource": {
         "BucketARN": "arn:aws:s3:::MyS3Bucket",
         "FileKey": "TickerReference.csv"
      },
      "TableName": "string"
   }
}
```

AddApplicationVpcConfiguration

AddApplicationVpcConfiguration 操作的以下示例请求代码将 VPC 配置添加到现有应用程序中：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
   }
}
```

CreateApplication

请求代码的以下示例请求代码：CreateApplication操作创建 Kinesis Data Analytics 应用程序：

```
{
  "ApplicationName":"MyApplication",
  "ApplicationDescription":"My-Application-Description",
  "RuntimeEnvironment":"FLINK-1_13",
  "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
  "CloudWatchLoggingOptions":[
    {
      "LogStreamARN":"arn:aws:logs:us-east-1:123456789123:log-group:my-log-group:log-
stream:My-LogStream"
    }
  ],
  "ApplicationConfiguration": {
    "EnvironmentProperties":
      {"PropertyGroups":
        [
          {"PropertyGroupId": "ConsumerConfigProperties",
            "PropertyMap":
              {"aws.region": "us-east-1",
              "flink.stream.initpos": "LATEST"}
          },
          {"PropertyGroupId": "ProducerConfigProperties",
            "PropertyMap":
              {"aws.region": "us-east-1"}
          },
        ]
      },
    "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
```

```
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType":"ZIPFILE"
    },
      "FlinkApplicationConfiguration":{
      "ParallelismConfiguration":{
        "ConfigurationType":"CUSTOM",
        "Parallelism":2,
        "ParallelismPerKPU":1,
        "AutoScalingEnabled":true
      }
    }
  }
}
```

CreateApplication快照

请求代码的以下示例请求代码：CreateApplication快照操作创建应用程序状态快照：

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MySnapshot"
}
```

DeleteApplication

请求代码的以下示例请求代码：DeleteApplication操作删除 Kinesis Data Analytics 应用程序：

```
{"ApplicationName": "MyApplication",
"CreateTimestamp": 12345678912}
```

DeleteApplicationCloudWatchLoggingOption

请求代码的以下示例请求代码：DeleteApplicationCloudWatchLoggingOption操作删除亚马逊 CloudWatchKinesis Data Analytics 应用程序的日志记录选项：

```
{
    "ApplicationName": "MyApplication",
    "CloudWatchLoggingOptionId": "3.1"
    "CurrentApplicationVersionId": 3
}
```

DeleteApplicationInputProcessing配置

请求代码的以下示例请求代码：DeleteApplicationInputProcessing配置操作从 Kinesis Data Analytics 应用程序中删除输入处理配置：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "InputId": "2.1"
}
```

DeleteApplication输出

请求代码的以下示例请求代码：DeleteApplication输出操作从 Kinesis Data Analytics 应用程序中删除应用程序输出：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "OutputId": "4.1"
}
```

DeleteApplicationReferenceData源

请求代码的以下示例请求代码：DeleteApplicationReferenceData源操作从 Kinesis Data Analytics 应用程序中删除应用程序引用数据源：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5,
   "ReferenceId": "5.1"
}
```

DeleteApplication快照

请求代码的以下示例请求代码：DeleteApplication快照操作删除应用程序状态快照：

```
{
   "ApplicationName": "MyApplication",
   "SnapshotCreationTimestamp": 12345678912,
   "SnapshotName": "MySnapshot"
}
```

DeleteApplicationVpcConfiguration

DeleteApplicationVpcConfiguration 操作的以下示例请求代码从应用程序中删除现有的 VPC 配置：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfigurationId": "1.1"
}
```

DescribeApplication

请求代码的以下示例请求代码：DescribeApplication操作返回 Kinesis Data Analytics 应用程序的详细信息：

```
{"ApplicationName": "MyApplication"}
```

DescribeApplication快照

请求代码的以下示例请求代码：DescribeApplication快照操作返回有关应用程序状态快照的详细信息：

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MySnapshot"
}
```

DiscoverInput架构

请求代码的以下示例请求代码：DiscoverInput架构操作从流式传输源中生成架构：

```
{
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "arn:aws:lambda:us-east-1:012345678901:function:MyLambdaFunction"
      }
   },
   "InputStartingPositionConfiguration": {
      "InputStartingPosition": "NOW"
   },
   "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/ExampleInputStream",
   "S3Configuration": {
      "BucketARN": "string",
      "FileKey": "string"
   },
   "ServiceExecutionRole": "string"
}
```

请求代码的以下示例请求代码：DiscoverInput架构操作从引用源中生成架构：

```
{
   "S3Configuration": {
      "BucketARN": "arn:aws:s3:::mybucket",
      "FileKey": "TickerReference.csv"
   },
   "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

ListApplications

请求代码的以下示例请求代码：ListApplications操作返回您账户中 Kinesis Data Analytics 应用程序的列表：

```
{
```

```
   "ExclusiveStartApplicationName": "MyApplication",
   "Limit": 50
}
```

ListApplication快照

请求代码的以下示例请求代码：ListApplication快照操作返回应用程序状态快照列表：

```
{"ApplicationName": "MyApplication",
   "Limit": 50,
   "NextToken": "aBcDeFgHiJkLmNoPqRsTuVwXyZ0123"
}
```

StartApplication

请求代码的以下示例请求代码：StartApplication操作启动 Kinesis Data Analytics 应用程序，并从最新快照（如有）中加载应用程序状态：

```
{
    "ApplicationName": "MyApplication",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

StopApplication

请求代码的以下示例请求代码：API_StopApplication操作停止 Kinesis Data Analytics 应用程序：

```
{"ApplicationName": "MyApplication"}
```

UpdateApplication

请求代码的以下示例请求代码：UpdateApplication操作更新 Kinesis Data Analytics 应用程序以更改应用程序代码的位置：

```
{"ApplicationName": "MyApplication",
"CurrentApplicationVersionId": 1,
"ApplicationConfigurationUpdate": {
      "ApplicationCodeConfigurationUpdate": {
         "CodeContentTypeUpdate": "ZIPFILE",
         "CodeContentUpdate": {
            "S3ContentLocationUpdate": {
               "BucketARNUpdate": "arn:aws:s3:::my_new_bucket",
               "FileKeyUpdate": "my_new_code.zip",
               "ObjectVersionUpdate": "2"
         }
```

```
        }
    }
}
```

Kinesis Data Analytics Apache Flink API 参考

有关 Kinesis Data Analytics Apache Flink 提供的 API 的信息，请参阅Kinesis Data Analytics API 参考.