Amazon Kinesis Data Analytics

Amazon Kinesis Data Analy

Amazon Kinesis Data Analytics: Amazon Kinesis Data Analy

Table of Contents

AAmazon Kinesis Data Analytics Flink

借助适用于 Apache Flink 的 Amazon Kinesis 数据分析，你可以使用 Java、Scala 或 SQL 来处理和分析流数据。该服务使您能够针对流媒体源编写和运行代码，以执行时间序列分析、提供实时仪表板和创建实时指标。

你可以使用基于 A pache Fl ink 的开源库在 Kinesis Data Analytics 中构建 Java 和 Scala 应用程序。Apache Flink 是处理数据流的常用框架和引擎。

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

Kinesis Data Analytics 为您的 Apache Flink 应用程序提供底层基础架构。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）。你可以使用高级的 Flink 编程功能（例如运算符、函数、源代码和接收器），就像你自己托管 Flink 基础架构时使用它们一样。

入门

您可以先创建一个持续读取和处理流数据的 Kinesis Data Analytics 应用程序。然后，使用您选择的 IDE 编写代码，并使用实时流数据对其进行测试。您还可以配置希望 Kinesis Data Analytics 将结果发送到的目的地。

首先，我们建议您阅读以下章节：

- Kinesis Data Analytics for Apache Flink (p. 2)
- Amazon Kinesis for Apache Flink (DataStream API) (p. 87)

Kinesis Data Analytics for Apache Flink

Kinesis Data Analytics for Apache Flink 是一项完全托管式服务，可让您使用 Apache Flink 应用程序处理流式传输数据。

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。你在本地编写和构建 Apache Flink 应用程序。

应用程序主要使用 DataStream API 或表 API。其他 Apache Flink API 也可供你使用，但它们在构建流媒体应用程序中不太常用。

这两个 API 的特点如下：

DataStream API

Apache Flink DataStream API 编程模型基于两个组件：

- 数据流：连续数据记录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 DataStream API 创建的应用程序执行以下操作：

- 从数据源（例如 Kinesis 直播或亚马逊 MSK 主题）读取数据。
- 对数据进行转换，例如过滤、聚合或丰富。
- 将转换后的数据写入数据接收器。

使用 DataStream API 的应用程序可以用 Java 或 Scala 编写，并且可以从 Kinesis 数据流、亚马逊 MSK 主题或自定义源中读取。

您的应用程序使用连接器处理数据。Apache Flink 使用以下连接器：

- 来源：用于读取外部数据的连接器。
- Sin@@ k：用于写入外部位置的连接器。
- 操作员：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。

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

表 API

Apache Flink API 编程模型基于以下组件：

- 表环境：基础数据的接口，用于创建和托管一个或多个表。
- 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 Amazon MSK 主题。
- 表函数：用于转换数据的 SQL 查询或 API 调用。
- T@@ able Sink：用于将数据写入外部位置，如 Amazon S3 存储桶。

使用 Table API 创建的应用程序执行以下操作：

- `TableEnvironment`通过连接到 a 来创建`Table Source`。
- `TableEnvironment`使用 SQL 查询或表 API 函数在中创建表。
- 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 Table API 的应用程序可以用 Java 或 Scala 编写，并且可以使用 API 调用或 SQL 查询查询数据。

有关使用 Table API 的更多信息，请参阅表 API (p. 19)。

创建您的 Kinesis Data Analytics

Kinesis Data Analytics 应用程序是由 Kinesis Data Analytics 服务托管的Amazon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. 25)：您可以向应用程序提供的参数。无需重新编译应用程序代码即可更改这些参数。
- 容错能力 (p. 27)：您的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. 268)：您的应用程序如何将事件记录到 CloudWatch 日志。
- 扩缩 (p. 33)：您的应用程序如何配置计算资源。

您可以使用控制台或创建 Kinesis Data Analytics 应用程序Amazon CLI。要开始创建 Kinesis Data Analytics 应用程序，请参阅入门指南 (DataStream API) (p. 87)。

Kinesis Data Analytics for Apache Flin

本主题包含有关创建Amazon Kinesis Data Analytics for Apache Flink 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 支持的最新版本的 Apache Flink 是 1.15.2。有关升级 Kinesis Data Analytics 应用程序的信息，请参阅升级应用程序 (p. 325)。

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 支持大小不超过 512 MB 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。

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 运行时的 Kinesis Data Analytics (aws-kinesisanalytics-runtime)	1.2.0
AmazonKinesis 连接器 (flink-connector-kinesis)	1.15.2
Apache Beam（仅限光束应用程序）	2.33.0，Jackson 版本 2.12.2

有关使用 Apache Flink 版本 1.15.2 的 Kinesis Data Analytics 应用程序`pom.xml`的文件示例，请参阅 Kinesis Data Analytics 入门应用程序。

有关创建使用 Apache Beam Data Analytics 应用程序的信息，请参阅，请参阅，请参阅，请参使用Apache Beam (p. 136)阅，请参

指定应用程序的 Apache Flink 版本

使用适用于 Flink Runtime 版本 1.1.0 及更高版本的 Kinesis Data Analytics 时，您需要指定应用程序在编译应用程序时使用的 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=1.15.3
```

有关使用旧版本的 Apache Flink 构建应用程序的信息，请参阅较早的版本 (p. 323)。

创建您的 Kinesis Data Analytics

生成应用程序代码后，您可以执行以下操作来创建 Kinesis Data Analytics 应用程序：

- 上传您的应用程序代码：将您的应用程序代码上传到 Amazon S3 存储桶，将您的应用程序上传到 Amazon S3 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，请参阅入门指南 (DataStream API) (p. 87)教程中的the section called “上传 Apache Flink 流式处理 Java 代码” (p. 92)。
- 创建您的 Kinesis Data Analytics 应用程序：使用以下方法之一创建 Kinesis Data Analytics 应用程序：
  - 使用Amazon控制台创建 Kinesis Data Analytics 应用程序：您可以使用Amazon控制台创建和配置应用程序。

    当您使用控制台创建应用程序时，将为您创建应用程序的依赖资源（例如 CloudWatch 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，请参阅入门指南 (DataStream API) (p. 87)教程中的the section called “创建并运行应用程序（控制台）” (p. 93)。

- 使用Amazon CLI 创建 Kinesis Data Analytics 应用程序：您可以使用Amazon CLI 创建和配置您的应用程序。

  使用 CLI 创建应用程序时，还必须手动创建应用程序的依赖资源（例如 CloudWatch 日志流、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，请参阅入门指南 (DataStream API) (p. 87)教程中的the section called “使用 CLI 创建和运行应用程序” (p. 96)。

  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

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，即可启动应用程序。启动 Kinesis Data Analytics 通常需要需要需要需要需要几需要需要几

可以使用以下方法之一以启动应用程序：

- 使用Amazon控制台启动 Kinesis Data Analytics 应用程序：您可以通过在控制Amazon台的应用程序页面上选择 “运行” 来运行应用程序。
- 使用Amazon API 启动您的 Kinesis Data Analytics 应用程序：您可以使用StartApplication操作运行应用程序。

正在验证您的 Kinesis Data Analytics

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 Logs 和 L CloudWatch ogs Insights 来验证您的应用程序是否正常运行。有关在 Kinesis Dat CloudWatch a Analytics 应用程序的信息，请参阅，请参阅，请参阅，请参日志记录和监控 (p. 268)阅，
- 使用 CloudWatch 指标：您可以使用 CloudWatch 指标监控应用程序的活动或应用程序用于输入或输出的资源（例如 Kinesis 流、Kinesis Data Firehose 传输流或 Amazon S3 存储桶）中的活动。有关 CloudWatch 指标的更多信息，请参阅亚马逊 CloudWatch 用户指南中的使用指标。
- 监控输出位置：如果您的应用程序将输出写入某个位置（例如 Amazon S3 存储桶或数据库），则可以监控该位置以获取写入数据。

Kinesis Data Analytics for Apache Flin

本主题包含运行Amazon Kinesis Data Analytics for Apache Flink。

当你运行 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 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。Job 管理器将应用程序的执行分成任务。每项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的性能，也可以检查整个Job 管理器的性能。

有关 Apache Flink 任务的信息，请参阅 A pache Flink 文档中的任务和日程安排。

申请和Job 状态

您的应用程序和应用程序的任务均处于当前执行状态：

- 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：
  - 稳定的应用程序状态：在您更改状态之前，您的应用程序通常会保持以下状态：
    - 就绪：新的或已停止的应用程序在运行之前处于“就绪”状态。
    - 正在运行：成功启动的应用程序处于“正在运行”状态。
  - 临时应用程序状态：处于这些状态的应用程序通常正在过渡到另一种状态。如果应用程序在一段时间内保持临时状态，则可以在Force参数设置为的情况下使用StopApplication操作来停止应用程序true。这些状态包括：
    - STARTING:在StartApplication动作之后发生。应用程序正在从状态过渡READY到RUNNING状态。
    - STOPPING:在StopApplication动作之后发生。应用程序正在从状态过渡RUNNING到READY状态。
    - DELETING:在DeleteApplication动作之后发生。该应用程序正在删除 Application（删除）
    - UPDATING:在UpdateApplication动作之后发生。应用程序正在更新，并将转换回RUNNING或READY状态。
    - AUTOSCALING:应用程序的AutoScalingEnabled属性 ParallelismConfiguration设置为true，该服务正在提高应用程序的并行度。当应用程序处于此状态时，您可以使用的唯一有效 API StopApplication操作是Force参数设置为的操作true。有关自动扩展的信息，请参阅自动扩展 (p. 35)。
    - FORCE_STOPPING:在Force参数设置为的情况下调用StopApplication操作后发生true。该应用程序正在强制停iew 应用程序从STARTING、UPDATINGSTOPPING、或AUTOSCALING状态过渡到READY状态。
    - ROLLING_BACK:在调用RollbackApplication操作后发生。该应用程序正在回滚到以前的版本。应用程序从UPDATING或AUTOSCALING状态过渡到RUNNING状态。
    - ROLLED_BACK:成功回滚应用程序后，这将成为您回滚的版本的状态。有关回滚应用程序的信息，请参阅RollbackApplication。
    - MAINTENANCE:在 Kinesis Data Analytics 向您的应用程序应用补丁时发生。有关更多信息，请参阅维护 (p. 307)：

  您可以使用控制台或使用DescribeApplication操作来检查应用程序的状态。
- Job 状态：当您的申请处于RUNNING状态时，您的任务的状态描述了其当前的执行阶段。任务以CREATED状态开始，然后在开始时继续进入RUNNING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，您的应用程序会进入RESTARTING状态。
  - 对于使用 Apache Flink 1.8 及更早版本的应用程序，您的应用程序将进入FAILING状态。

  然后，应用程序会进入RESTARTING或FAILED状态，具体取决于是否可以重新启动作业。

  您可以通过检查应用程序 CloudWatch 日志的状态更改来检查作业的状态。

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 作业变为“完成”状态时，Kinesis Data Analytics 应用程序的状态设置为“就绪”。有关 Flink 作业状态的更多信息，请参阅任务和调度。

应用程序资源

本节介绍您的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帮助您设计、创建和维护高性能和稳定的 Kinesis Data Analytics 应用程序。

Kinesis Data Analytics 应用程序

Kinesis Data Analytics 是一项为托管 Apache Flink 应用程序创建环境的Amazon服务。Kinesis Data Analytics 服务使用名为 Kinesis 处理单元 (KPU) 的单位提供资源。

一个 KPU 代表以下系统资源：

- 一个 CPU 核心
- 4 GB 内存，其中 1 GB 为本机内存，3 GB 为堆内存
- 50 GB 的磁盘空间

KPU 以不同的执行单元（称为任务和子任务）运行应用程序。您可以将子任务视为等同于线程。

应用程序可用的 KPU 数量等于应用程序的`Parallelism`设置除以应用程序的`ParallelismPerKPU`设置。

有关应用程序并行的更多信息，请参阅扩缩 (p. 33)。

Apache Flink 应用程序资源

Apache Flink 环境使用称为任务槽的单元为您的应用程序分配资源。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 任务槽分配给一个 KPU。分配给单个 KPU 的插槽数等于应用程序的`ParallelismPerKPU`设置。有关任务槽的更多信息，请参阅 Apache Flink 文档中的Jo b 调度。

操作员并行性

您可以设置操作员可以使用的最大子任务数。此值称为运算符并行度。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。

您可以使用该`setParallelism`方法设置应用程序中运算符的并行度。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。

有关运算符链接的更多信息，请参阅 Apache Flink 文档中的任务链和资源组。

操作员创建

通常，每个运算符使用单独的子任务来执行，但是如果多个运算符总是按顺序执行，则运行时可以将它们全部分配给同一个任务。此过程称为操作员链接。

如果多个顺序运算符都对相同的数据进行操作，则可以将它们链接到单个任务中。以下是一些示例方案：

- 运营商进行一对一的简单转发。
- 所有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，它可以节省系统资源，因为该服务不需要执行网络操作并为每个操作员分配子任务。要确定您的应用程序是否使用运算符链接，请查看 Kinesis Data Analytics 控制台中的任务图。应用程序中的每个顶点代表一个或多个运算符。该图显示了已链接为单个顶点的运算符。

DataStream API

你的 Apache Flink 应用程序使用 Apache Flink DataStream API 来转换数据流中的数据。

本节包含以下主题：

- 使用 AP DataStream I 在 Apache Flink 的 Kinesis Data Analytics 中使用连接器移动数据 (p. 8)：这些组件在您的应用程序和外部数据源和目标之间移动数据。
- Data Kinesis Data Analytics for Apache Flink DataStream (p. 18)：这些组件转换或分组应用程序中的数据元素。
- 使用 Apache FlKinesis Data Analytics for Apache Flink DataStream (p. 19)：本主题介绍使用 DataStream API 时 Kinesis Data Analytics 如何跟踪事件。

使用 AP DataStream I 在 Apache Flink 的 Kinesis Data Analytics 中使用连接器移动数据

在适用于 Apache Flink DataStream API 的 Amazon Kinesis Data Analytics 中，连接器是将数据移入和移出 Kinesis 数据分析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。连接器由用于与亚马逊服务和第三方系统交互的完整模块组成。

连接器类型包括：

- 源 (p. 14)：从 Kinesis 数据流、文件或其他数据源向应用程序提供数据。
- 接收器 (p. 15)：将数据从您的应用程序发送到 Kinesis 数据流、Kinesis Data Firehose 传输流或其他数据目的地。
- 异步 I/O (p. 18)：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 Apache Flink 文档中的连接器。

> Warning
>
> 如果您在 Flink 1.6、1.8、1.11 或 1.13 上运行应用程序，并且想要在中东（阿联酋）或亚太地区（雅加达）地区运行，则可能需要使用更新的连接器重建应用程序存档或升级到 Flink 1.15。以下是推荐的指导方针：
>
> 连接器升级

Fli 版本	使用的连接器	解析
1.6 1.8	Firehose	您的应用程序依赖于 Firehose

Fli 版本	使用的连接器	解析
		连接器的过时版本，该版本不知道更新的 Amazon 区域。Powe 连接器连接器版本 2. v2.1.0

Fli 版本	使用的连接器	解析
1.8	Kinesis	您的应用程序依赖于 Flink Kinesis 连接器的过时版本，该版本不知道更新的 Amazon 区域。使用 Flink Kinesis 连接器版本 1.6.1 重建您的应用程序存档。

Fli 版本	使用的连接器	解析
		https://github.com/awslabs/amazon-kinesis-connector-flink / tree/1.6.1

Fli 版本	使用的连接器	解析
1.11	Kinesis	您的应用程序依赖于 Flink Kinesis 连接器的过时版本，该版本不知道更新的 Amazon 区域。使用 Flink 连接器连接器版本 2。 https://github.com/awslabs/amazon-kinesis-connector-flink/tree/2.4.1

Fli 版本	使用的连接器	解析
1.6 和 1.13	Kinesis	您的应用程序依赖于 Flink Kinesis 连接器的过时版本，该版本不知道更新的 Amazon 区域。不幸的是，Flink 不再发布 1.6/1.13 连接器的补丁或错误修复。我们

Fli 版本	使用的连接器	解析
		建议通过使用 Flink 1.15 重建应用程序存档来更新到 Flink 1.15。

为Data Analytics for Apache Flink Powe Flink

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。本节描述了可用于亚马逊服务的来源。

Kinesis Data Streams

该`FlinkKinesisConsumer`源从 Amazon Kinesis 数据流向您的应用程序提供流式数据。

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. 91)。

创建使用`FlinkKinesisConsumer` EFO 消费者的

FlinkKinesisConsumer 现在支持增强型扇出 (EFO)。

如果 Kinesis 使用者使用 EFO，则 Kinesis Data Streams 服务会为其提供自己的专用带宽，而不是让使用者与其他使用者共享直播的固定带宽。

有关对 Kinesis 消费者使用 EFO 的更多信息，请参阅 FLIP-128：增强版Amazon Kinesis 消费者风扇输出功能。

您可以通过在 Kinesis 使用者上设置以下参数来启用 EFO 使用者：

- RECORD_PUBLISHER_TYPE：将此参数设置为 E FO，让您的应用程序使用 EFO 使用者访问 Kinesis 数据流数据。
- EFO_CONSUMER_NAME：将此参数设置为在该直播的使用者中唯一的字符串值。在同一 Kinesis Data Stream 中重复使用消费者名称将导致先前使用该名称的使用者被终止。

要将 a 配置`FlinkKinesisConsumer`为使用 EFO，请向使用者添加以下参数：

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

有关使用 EFO 使用者的 Kinesis Data Analytics 应用程序的示例，请参阅EFO 消费者 (p. 171)。

Amazon MSK

该`KafkaSource`来源向您的应用程序提供来自 Amazon MSK 主题的流式传输数据。

创建`KafkaSource`

以下代码示例说明了如何创建 `KafkaSource`：

```
KafkaSource<String> source = KafkaSource.<String>builder()
    .setBootstrapServers(brokers)
    .setTopics("input-topic")
    .setGroupId("my-group")
    .setStartingOffsets(OffsetsInitializer.earliest())
    .setValueOnlyDeserializer(new SimpleStringSchema())
    .build();

env.fromSource(source, WatermarkStrategy.noWatermarks(), "Kafka Source");
```

有关使用 `KafkaSource` 的更多信息，请参阅MSK 复制 (p. 167)。

在Data Analytics for Apache Flink Powe Flink

在应用程序代码中，您可以使用 Apache Flink 接收器将数据从 Apache Flink 流写入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。以下水槽可用于Amazon：

Kinesis Data Streams

Apache Flink 在 Apache Fl ink 文档中提供了有关 Kinesis Data Streams 连接器的信息。

有关使用 Kinesis 数据流进行输入和输出的应用程序的示例，请参见入门指南 (DataStream API) (p. 87)。

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. 158)。

Kinesis Data Firehose

FlinkKinesisFirehoseProducer是一款可靠、可扩展的 Apache Flink 接收器，用于使用 Kinesis Data Firehos e 服务存储应用程序输出。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

以下代码示例演示如何创建和配置 Apache Flink 数据流FlinkKinesisFirehoseProducer并将数据从 Kinesis Data Firehose 服务发送到 Kinesis Data Firehose 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 水槽的完整教程，请参阅the section called “Kinesis Data Firehose” (p. 178)。

在Data Analytics for Apache Flink pache F

异步 I/O 运算符使用数据库等外部数据源丰富流数据。Kinesis Data Analytics 异步丰富了直播事件，因此可以批处理请求以提高效率。

有关更多信息，请参阅 Apache Flink 文档中的异步 I/O。

Data Kinesis Data Analytics for Apache Flink DataStream

要转换适用于 Apache Flink 的 Amazon Kinesis 数据分析中的传入数据，你可以使用 Apache Flink 运算符。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 Apache Flink 文档中的运算符。

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，请参阅入门指南 (DataStream API) (p. 87)。入门应用程序的源代码可在 Kinesis Data Analytics Java 示例 GitHub 存储库的 “入门” 中找到。

使用 Apache FlKinesis Data Analytics for Apache Flink DataStream

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 Apache Flink 文档中的事件时间。

表 API

你的 Apache Flink 应用程序使用 Apache Flink Table API 使用关系模型与流中的数据进行交互。您可以使用 Table API 访问表源的数据，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 连接连接器和连接 (p. 19)：这些组件在您的应用程序和外部数据源和目标之间移动数据。
- 表 API 时间属性 (p. 20)：本主题介绍使用 Table API 时 Kinesis Data Analytics 如何跟踪事件。

表 API 连接连接器和连接

在 Apache Flink 编程模型中，连接器是应用程序用来从外部来源（例如其他Amazon服务）读取或写入数据的组件。

通过 Apache Flink Table API，你可以使用以下类型的连接器：

- 表 API 源代码 (p. 19)：您可以使用 Table API 源连接器使用 API 调`TableEnvironment`用或 SQL 查询在自己的数据库中创建表。
- 表 API 接收器 (p. 20)：您可以使用 SQL 命令将表数据写入外部来源，例如 Amazon MSK 主题或 Amazon S3 存储桶。

表 API 源代码

您可以从数据流创建表源。以下代码基于 Amazon MSK 主题创建表：

```
//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 Apache Flink 文档中的表和连接器。

表 API 接收器

要将表数据写入接收器，请在 SQL 中创建接收器，然后在该`StreamTableEnvironment`对象上运行基于 SQL 的接收器。

以下代码示例演示了如何将表数据写到 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");
```

你可以使用`format`参数控制 Kinesis Data Analytics 使用什么格式将输出写入接收器。有关格式的信息，请参阅 Apache Flink 文档中的格式。

有关表格接收器的更多信息，请参阅 Apache Flink 文档中的表格和连接器。

用户定义的源头和汇点

您可以使用现有的 Apache Kafka 连接器向其他Amazon服务（例如亚马逊 MSK 和Amazon S3）发送数据。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点。有关更多信息，请参阅 Apache Flink 文档中的用户定义源和接收器。

表 API 时间属性

数据流中的每条记录都有多个时间戳，用于定义与记录相关的事件发生的时间：

- 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。
- 摄取时间：您的应用程序从数据流中检索记录的时间。
- 处理时间：您的应用程序处理记录的时间。

当 Apache Flink Table API 根据记录时间创建窗口时，您可以使用setStreamTime特征方法定义它使用哪个时间戳。

有关在 Table API 中使用时间戳的更多信息，请参阅 Apache Flink 文档中的时间属性。

在Kinesis Data Analytics

Apache Flink 版本 1.15.2 包括支持使用 Python 版本 3.8 使用PyFlink库创建应用程序。您可以通过执行以下操作使用 Python 创建 Kinesis Data Analytics 应用程序：

- 使用`main`方法将您的 Python 应用程序代码创建为文本文件。
- 将您的应用程序代码文件和任何 Python 或 Java 依赖项捆绑到一个 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，指定您的 Amazon S3 代码位置、应用程序属性和应用程序设置。

从较高的层面上讲，Python Table API 是围绕 Java 表 API 的包装。有关 Python 表 API 的信息，请参阅 Apache Flink 文档中的 Pyth on 表 API 简介。

为 Python 应用程序编程 Kinesis Data Analytics

你可以使用 Apache Flink Python 表 API 对适用于 Python 的 Kinesis Data Analytics 应用程序进行编码。Apache Flink 引擎将 Python Table API 语句（在 Python 虚拟机中运行）转换为 Java Table API 语句（在 Java 虚拟机中运行）。

您可以通过以下步骤使用 Simple Storage Storage Storage S

- 创建对的引用`StreamTableEnvironment`。
- 通过对`StreamTableEnvironment`引用执行查询，从源流数据创建`table`对象。
- 对您的`table`对象执行查询以创建输出表。
- 使用将输出表写入目的地`StatementSet`。

要开始在 Kinesis Data Analytics 中使用 Python 表 API，请参阅Amazon Kinesis Data Analytics for Apache (p. 113)。

读取和写入流数据

要读取和写入流数据，您需要在表环境中执行 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 查询会创建一个与 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)
```

读取流媒体数据

以下代码示例演示如何在表环境引用上面的`CreateTable` SQL 查询来读取数据：

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

写入流媒体数据

以下代码示例演示如何使用`CreateTable`示例中的 SQL 查询来创建输出表引用，以及如何使用 a 与表交互`StatementSet`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来配置应用程序，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为适用于 Java 的 Kinesis Data Analytics 应用程序指定应用程序属性的方法相同。您可以通过下列方式指定运行时属性：

- 使用动CreateApplication作。
- 使用动UpdateApplication作。
- 使用控制台配置应用程序。

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

以下代码示例演示了从`application_properties.json`文件中读取应用程序属性：

```
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"]
```

以下代码示例演示了从前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，将代码文件和依赖项捆绑到一个 zip 文件中。

您的 zip 文件必须包含带有`main`方法的 python 脚本，并且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。您无法为应用程序引用其他来源的库。

创建你的 Python Kinesis Data Analytics 应用程序

指定您的代码文件

创建应用程序的代码包后，您可以将其上传到 Amazon S3 存储桶。然后，您可以使用控制台或CreateApplication操作创建应用程序。

使用CreateApplication操作创建应用程序时，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：一个包含 Python 主方法的文本文件。
- jarfile：一个包含 Java 用户定义函数的 Java JAR 文件。
- PyF iles：一个 Python 资源文件，其中包含应用程序要使用的资源。
- PyA rchives：包含应用程序资源文件的 zip 文件。

有关 Apache Flink Python 代码文件类型的更多信息，请参阅 Apache Flink 文档中的命令行用法。

> Note
>
> Kinesis Data Analytics 不支持`pyModulepyExecutable`、或`pyRequirements`文件类型。所有代码、要求和依赖关系都必须位于您的 zip 文件中。你无法使用 pip 指定要安装的依赖关系。

以下 json 代码段示例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"
```

```
        }
      },
```

监控你的 Python Kinesis Data Analytics 应用程序

您可以使用应用程序 CloudWatch 的日志来监视您的 Python Kinesis Data Analytics 应用程序。

Kinesis Data Analytics 会为 Python 应用程序记录以下消息：

- 使用`print()`应用程序的`main`方法写入控制台的消息。
- 使用`logging`软件包在用户定义的函数中发送的消息。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：

```
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`函数中抛出异常，它将出现在应用程序的日志中。

  以下示例演示了 Python 代码引发的异常的日志条目：

```
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，我们建议您仅在应用程序开发期间使用自定义日志消息。

使用 CloudWatch 见解查询日志

以下 CloudWatch Insights 查询会在执行应用程序主功能时搜索 Python 入口点创建的日志：

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

Kinesis Data Analytics Apache Flink

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，无法添加运行时属性。

更新 Kinesis Analytics 的运行时属性

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 选择您的 Kinesis Data Analytics。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 (p. 423)。

> 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_15",
    "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

如果您使用的密钥在属性组中没有相应的运行时属性，Kinesis Data Analytics 会将该键值对添加为新属性。如果您为属性组中的现有运行时属性使用密钥，Kinesis Data Analytics 会更新该属性值。

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");
```

通常，您可以通过传入Properties对象来配置 Apache Flink 源代码或接收器，而无需检索单个属性。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，请参阅入门指南 (DataStream API) (p. 87)。入门应用程序的源代码可在 Kinesis Data Analytics Java 示例 GitHub 存储库的 “入门” 中找到。

在 Kinesis Data Analytics for Apache Flink

Checkpoint是用于在适用于 Apache Flink 的 Amazon Kinesis Data Analytics 中实现容错的方法。检查点是正在运行的应用程序的 up-to-date 备份，用于从意外的应用程序中断或故障转移中立即恢复。

有关 Apache Flink 应用程序中检查点的详细信息，请参阅 Apache Flink 文档中的检查点。

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. 30)：

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快照进行配置 ApplicationRestoreConfiguration，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仅提供一次性处理语义。

在 Kinesis Data Analytics for Apache Flink

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— 表示是否在应用程序中启用了检查点。
- CheckpointInterval— 包含检查点（持久性）操作之间的时间（以毫秒为单位）。
- ConfigurationType— 将此值设置DEFAULT为以使用默认的检查点行为。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

  如果此值设置ConfigurationType为DEFAULT，则将使用前面的值，即使通过使用Amazon Command Line Interface、或通过使用应用程序代码中的值将其设置为其他值也是如此。

  Note

  对于 Flink 1.15 以后，适用于 Apache Flink 的 Kinesis Data Analytics 将在自动创建快照stop-with-savepoint期间使用，即应用程序更新、缩放或停止。
- MinPauseBetweenCheckpoints— 一个检查点操作结束与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 (p. 423)。

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_15",
   "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_15",
   "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

快照是 Apache Flink S avepo int 的 Kinesis Data Analytics 实现。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 Apache Flink 文档中的保存点。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
>
> 我们建议您的应用程序每天创建几次快照，以便使用正确的状态数据正确重启。快照的正确频率取决于应用程序的业务逻辑。频繁拍摄快照可以恢复更新的数据，但会增加成本并需要更多的系统资源。

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. 306)。如果您的应用程序达到快照限制，则手动创建快照将失败，并显示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StartApplication或UpdateApplication操作的ApplicationRestoreConfiguration参数。

本主题包含下列部分：

- 自动创建快照 (p. 30)
- 从包含不兼容状态数据的快照中还原 (p. 31)
- 快照 API 示例 (p. 31)

自动创建快照

如果在应用程序true中设置SnapshotsEnabled为，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，以提供完全一次的处理语义。 ApplicationSnapshotConfiguration

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。

Note

Kinesis Data Analytics 会在创建快照期间触发中间保存点。对于 Flink 版本 1.15 或更高版本，中间保存点不再产生任何副作用。请参见触发保存点

自动创建的快照具有以下特性：

- 快照由服务管理，但您可以使用 ListApplicationSnapshots操作查看快照。自动创建的快照计入您的快照上限。
- 如果您的应用程序超过快照限制，手动创建的快照将失败，但是 Kinesis Data Analytics 服务在更新、扩展或停止应用程序时仍会成功创建快照。在手动创建更多快照之前，必须使用 DeleteApplicationSnapshot操作手动删除快照。

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恢复，请true使用UpdateApplication操作FlinkRunConfiguration将的AllowNonRestoredState参数设置为。

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。
- 删除了操作符：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。
- 修改了操作符：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。有关从快照恢复的更多信息，请参阅 Apache Flink 文档中的保存点。使用 Apache Flink 版本 1.8 或更高版本的应用程序可以从具有不同架构的快照中恢复。无法还原使用 Apache Flink 版本 1.6 的应用程序。对于 two-phase-commit 接收器，我们建议使用系统快照 (SW) 而不是用户创建的快照 (CreateApplicationSnapshot)。

  对于 Flink，Kinesis Data Analytics 会在创建快照期间触发中间保存点。在 Flink 1.15 以后，中间保存点不再产生任何副作用。请参阅触发保存点。

如果您需要恢复与现有 savepoint 数据不兼容的应用程序，我们建议您通过将StartApplication操作ApplicationRestoreType参数设置为，跳过从快照中恢复SKIP_RESTORE_FROM_SNAPSHOT。

有关 Apache Flink 如何处理不兼容状态数据的更多信息，请参阅 Apache Flink 文档中的状态架构演变。

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 (p. 423)。

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。您可以使用以下任一方法获取该SnapshotCreationTimestampListApplicationSnapshots值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"
      }
   }
}
```

AKinesis Data Analytics he

您可以为 Amazon Kinesis Data Analytics for Apache Flink 配置任务的parallel 执行和资源分配，以实现扩展。有关 Apache Flink 如何调度任务的parallel 实例的信息，请参阅 Apache Flink 文档中的并行执行。

主题

配置应用程序并行度和 ParallelismPer KPU

您可以使用以下ParallelismConfiguration属性为 Kinesis Data Analytics 应用程序任务（例如从源读取或执行运算符）配置parallel 执行：

- Parallelism— 使用此属性设置 Apache Flink 应用程序的默认并行度。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`— 使用该属性设置应用程序在其Kinesis 处理单元（KPU）可以安排的 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。有关如何请求提高限制的说明，请参阅Service Quotas 中的 “请求提高限制”。

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

分配 Kinesis 处理单元

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

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

Kinesis Data Analytics 可快速为您的应用程序提供资源，以应对吞吐量或处理活动的峰值。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. 34)中所述。

应用程序的默认 KPU 限制为 32 个。有关如何申请提高此限额的说明，请参阅Service Quotas 中的 “申请提高限额”。

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 (p. 423)。

`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_15",
   "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 可弹性扩展应用程序的并行度，以适应大多数场景下源的数据吞吐量和操作员的复杂性。Kinesis Data Analytics 监控应用程序的资源 (CPU) 使用情况，并相应地弹性地向上或向下扩展应用程序的并行度：

- 当您的 CPU 使用率在 75% 或以上持续 15 分钟时，您的应用程序会向上扩展（提高并行度）。
- 当 CPU 使用率在六小时内保持在 10% 以下时，您的应用程序会向下扩展（降低并行度）。

Kinesis Data Analytics 不会将应用程序的`CurrentParallelism`价值降低到低于应用程序的`Parallelism`设置。

当 Kinesis Data Analytics 服务扩展您的应用程序时，它将处于`AUTOSCALING`状态。您可以使用 DescribeApplication或 ListApplications操作检查当前的申请状态。当该服务正在扩展您的应用程序时，您可以使用的唯一有效的 API 操作是 StopApplication将`Force`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Amazon账户需要为 Kinesis Data Analytics 预置的 KPU 付费，这取决于您的应用程序`parallelism`和`parallelismPerKPU`设置。活动激增会增加您的 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。
- 扩展不适用于 Studio 笔记本电脑。但是，如果您将 Studio 笔记本部署为具有持久状态的应用程序，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. 306)：
- 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
  - 使用UpdateApplication操作配置应用程序parallelismPerKPU的parallelism和。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. 34)。
  - 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. 278)。

最大并行度注意事项

- AutoScale 逻辑可以防止将 Flink 作业扩展到并行度，从而干扰作业和操作员maxParallelism。例如，如果一个简单作业只有一个源和一个汇点，其中源有maxParallelism 16 个，汇点sink有 8 个，我们就不会自动将作业缩放到 8 以上。
- 如果没有maxParallelism为任务设置，Flink 将默认为 128。因此，如果您认为作业需要以比 128 更高的并行度运行，则必须为应用程序设置该数字。
- 如果您希望看到作业自动缩放，但看不到，请确保您的maxParallelism值允许。

有关更多信息，请参阅 Apache Flink 的增强监控和自动扩展

有关示例，请参见 kda-flink-app-autoscaling。

使用标记

本节介绍如何向 Kinesis Data Analytics 应用程序添加键值元数据标签。这些标签可用于以下目的：

- 确定各个 Kinesis Data Analytics 应用程序的账单。有关更多信息，请参阅 Billing and Cost Man age ment 指南中的使用成本分配标签。
- 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Amazon Identity and Access Management 用户指南中的使用标签控制访问。
- 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

在使用CreateApplication操作tags参数创建应用程序时，您可以添加标签。

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TagResource操作向应用程序添加标签。您无法使用UpdateApplication操作向应用程序添加标签。

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

要列出现有标签，请使用ListTagsForResource操作。

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，请使用UntagResource操作。

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" ]
}
```

CloudFormation 与 Kinesis Data Analytics

以下练习展示了如何启动通过在同一堆栈中Amazon CloudFormation使用 Lambda 函数创建的 Flink 应用程序。

开始前的准备工作

在开始本练习之前，请按照使用 at 创建 Flink 应用程序的步骤Amazon CloudFormation进行操作AWS::KinesisAnalytics::Application。

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 启动应用程序 API。调用将由 Flink 应用程序创建后Amazon CloudFormation的事件触发。我们将在本练习的稍后部分讨论如何设置堆栈以触发 Lambda 函数，但首先我们将重点介绍 Lambda 函数声明及其代码。我们在这个例子中使用Python3.8运行时。

```
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)})
```

在前面的代码中，Lambda 将处理传入Amazon CloudFormation事件，过滤掉除`Create`和之外的所有内容`Update`，获取应用程序状态，如果状态为，则启动它`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

您为 Lambda 创建一个角色以成功与应用程序 “通信” 并写入日志。此角色将使用默认托管策略，但您可能需要使用自定义策略来缩小范围。

```
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 资源将在同一个堆栈中创建 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。这是使用自定义资源完成的。

```
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 应用程序所需要的一切。现在，您可以创建自己的堆栈了，或者使用下面的完整示例来看看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略微扩展版本，并通过模板参数RunConfiguration进行了额外调整。这是一个供您尝试的工作堆栈。请务必阅读随附注释：

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_15'
    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，您可能需要调整 Lambda 的角色以及应用程序本身。

在创建上面的堆栈之前，不要忘记指定您的参数。

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"
  }
]
```

YOUR_JAR用您的特定要求替换YOUR_BUCKET_ARN和。您可以按照本指南创建 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
```

你现在可以导航到 https://console.aws.amazon.com/cloudformation 查看进度。创建后，你应该会看到你的 Flink 应用程序处于`Starting`状态。可能需要几分钟的时间`Running`。

有关更多信息，请参阅下列内容：

- 使用Amazon CloudFormation（第 1 部分，共 3 部分）检索任何Amazon服务属性的四种方法。
- 演练：查找 Amazon 系统映像

使用 Amazon Kinesis Data Analytics for Apache

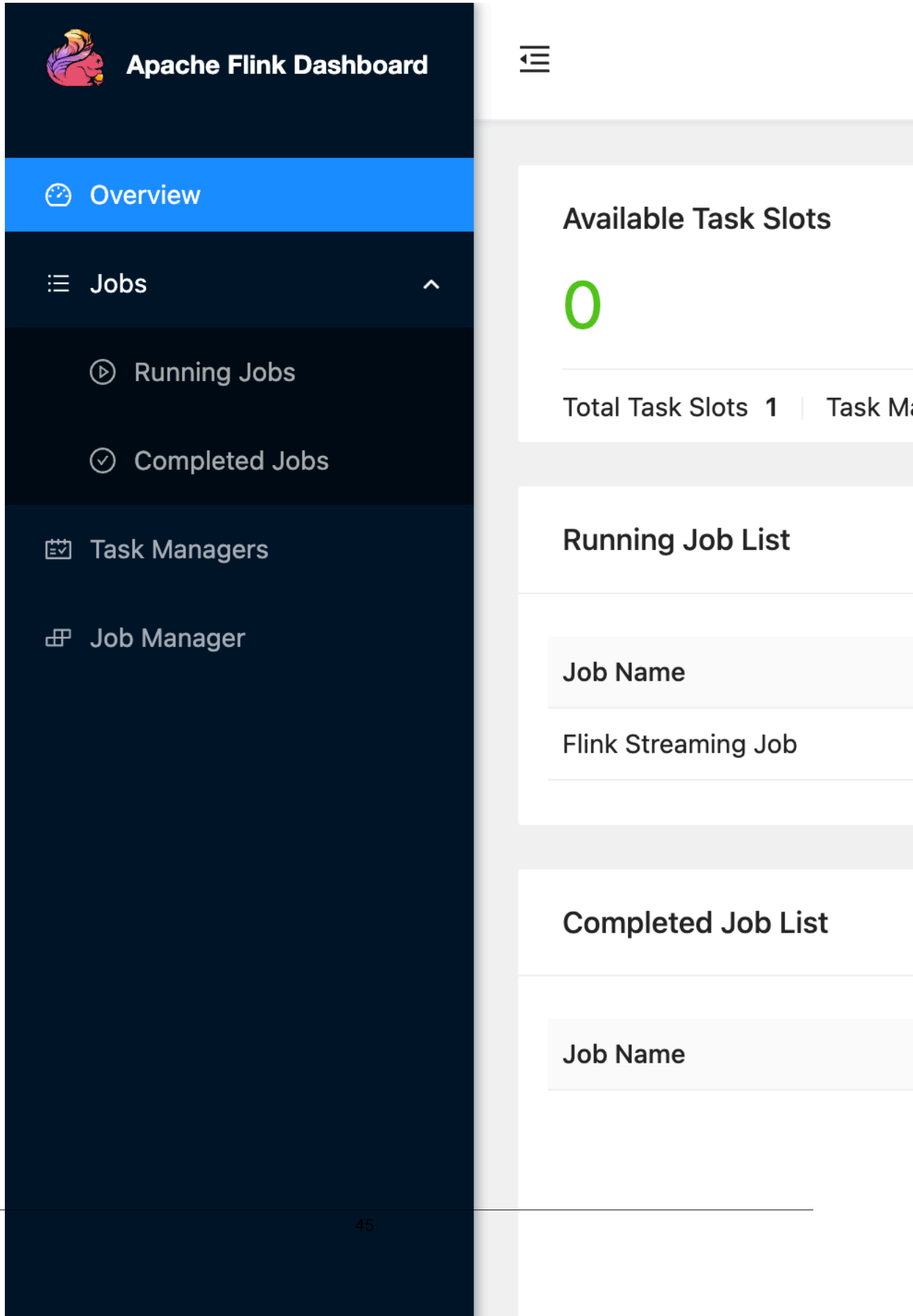
您可以使用应用程序的 Apache Flink 控制面板来监控 Kinesis Data Analytics 应用程序的运行状况。您的应用程序控制面板显示以下信息：

- 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、已取消和失败的作业。

有关 Apache Flink 任务管理器、任务槽和任务的信息，请参阅 Apache Flink 网站上的 Apache Flink 架构。

请注意以下关于在 Kinesis Data Analytics 应用程序中使用 Apache Flink 控制面板：

- 适用于 Kinesis Data Analytics 应用程序的 Apache Flink 仪表板是只读的。你无法使用 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 Internet Explorer 不兼容。

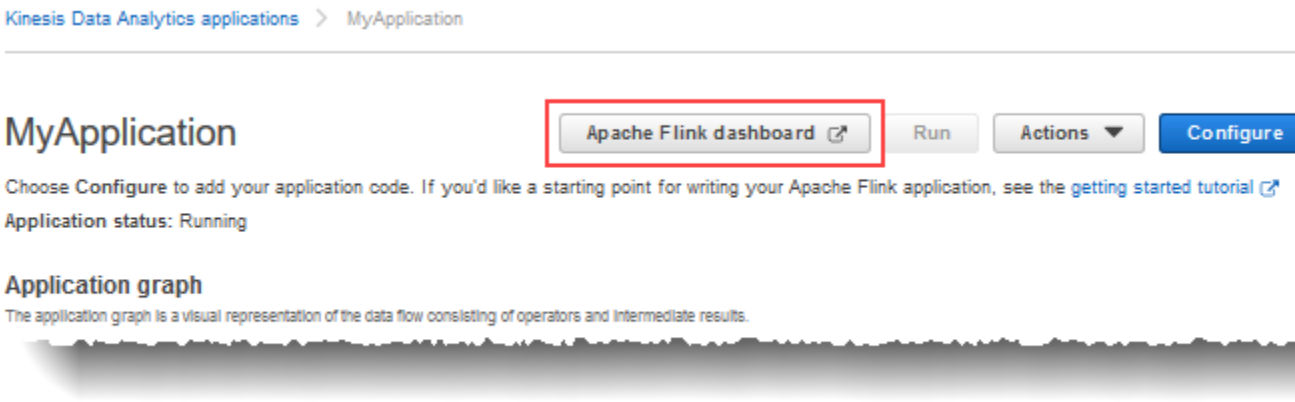

Apache Flink Dashboard
Overview
Jobs
Running Jobs
Completed Jobs
Task Managers
Job Manager
Available Task Slots
0
Total Task Slots 1
Task Ma
Running Job List
Job Name
Flink Streaming Job
Completed Job List
Job Name

访问应用程序的 Apache Flink 控制面板

您可以通过 Kinesis Data Analytics 控制台访问应用程序的 Apache Flink 控制台，也可以使用 CLI 请求安全 URL 端点。

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，请在应用程序页面上选择 Apache Flink 控制面板。



Note

当您从 Kinesis Data Analytics 控制台打开仪表板时，控制台生成的 URL 将在 12 小时内有效。

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 来访问您的应用程序控制面板。您生成的 URL 在指定的时间内有效。

Note

如果您未在三分钟内访问生成的 URL，它将不再有效。

您可以使用 CreateApplicationPresignedUrl操作生成仪表板网址。您可以为操作指定以下参数：

- 应用程序名称
- URL 生效的时间（以秒为单位）
- 您可以指定`FLINK_DASHBOARD_URL`为 URL 类型。

发行版

本主题包含有关每个版本的 Kinesis Data Analytics for Apache Flink 支持的功能和推荐的组件版本的信息。

Amazon Kinesis Data Analytics for Amazics for

Kinesis Data Analytics 支持 Apache 1.15.2 中的以下新功能

功能	描述	Apache FLIP 参考
异步接收器	一个用于构建异步目的地的Amazon贡献框架，允许开发人员以不到之前工作量的一半来构建自定义Amazon连接器。有关更多信息，请参阅通用异步基础接收器。	FLIP-171：Async Sink。
Kinesis Data Firehos	Amazon使用 Async 框架贡献了一个新的亚马逊 Kinesis Firehose Sink。	Amazon Kinesis Data Firehos
停止使用 Savepoint	Stop with Savepoint 可确保彻底停止操作，最重要的是为依赖它们的客户提供一次性语义支持。	FLIP-34：使用保存点终止/暂停 Job。
Scala 去耦器	用户现在可以利用任何 Scala 版本的 Java API，包括 Scala 3。客户需要在他们的 Scala 应用程序中捆绑他们选择的 Scala 标准库。	FLIP-28：长期目标是让 flink-table 不使用 Scala。
统一连接器指标	Flink 为任务、任务和操作员定义了标准指标。Kinesis Data Analytics将继续支持汇流和源指标，并在1.15numRestarts 中parallelfullRestarts 引入可用性指标。	FLIP-33：标准化连接器指标和FLIP-179：公开标准化操作员指标。
通过通过检查点检验已完成任务	此功能在 Flink 1.15 中默认启用，即使任务图的某些部分已完成所有数据的处理，也可以继续执行检查点，如果它包含有界（批处理）源，则可能会发生这种情况。	FLIP-147：任务完成后Support 检查点。

Amazon Kinesis Data Analytics 中发生了变化

Kinesis 连接器

- 如果应用程序使用不支持的 Kinesis Connector 版本（捆绑到应用程序 JAR 中），则适用于 Apache Flink 的 Kinesis Data Analytics 版本 1.15 将自动阻止应用程序启动或更新。升级到适用于 Apache Flink 的 Kinesis Data Analytics 版本 1.15 时，请确保你使用的是最新的 Kinesis Connector。

- 这适用于任何版本 1.15.2 或更高版本。Kinesis Data Analytics for Apache Flink 将不支持所有其他版本，因为它们可能会导致一致性问题或该`Stop with Savepoint`功能失败，从而阻止干净的停止/更新操作。

EFO 连接器

升级到适用于 Apache Flink 的 Kinesis Data Analytics 版本 1.15 时，请确保你使用的是最新的 EFO Connector，即任何版本 1.15.3 或更高版本。有关原因的更多信息，请参阅 FLINK-29324。

Scala 去耦合

从 Flink 1.15.2 开始，你需要在你的 Scala 应用程序中捆绑你选择的 Scala 标准库。

Kinesis Data Firehos

升级到适用于 Apache Flink 的 Kinesis Data Analytics 版本 1.15 时，请确保你使用的是最新的 Amazon Kinesis Data Firehose Sink。

Kafka 连接器

升级到适用于 Apache FIAmazon Kinesis Data Analytics 版本 1.15 时，请确保你使用的是最新的 Kafka 连接器 API。Apache Flink 已经过时 FlinkKafkaConsumer，FlinkKafkaProducer这些 Kafka sink 的 API 无法提交到 Flink 1.15 版的 Kafka。确保您正在使用KafkaSource和KafkaSink。

组件

组件	版本
Java	11（推荐）
Scala	2.12
Kinesis Data Analytics Flink 运行时 (aws-kinesisanalytics-runtime)	1.2.0
AmazonKinesis 连接器 (flink-connector-kinesis)	1.15.2
Apache Beam（仅限 Beam 应用程序	2.33.0，以 Jackson 版本 2.12.2

已知问题

异步接收器性能

众所周知，在高负载场景下，1.15 的 AsyncSink 性能与传统 sink 相比有所下降，尤其是在分片数量较多（64 或更多）的情况下。其他影响因素包括更大的负载大小和更高的并行度应用程序。

Kinesis Data Analytics 工作室

Studio 利用 Apache Zeppelin 笔记本电脑为开发、调试代码和运行 Apache Flink 流处理应用程序提供单接口开发体验。齐柏林飞艇的 Flink 解释器需要升级才能支持 Flink 1.15。这项工作是由 Zeppelin 社区安排的，我们将在完成后更新这些笔记。您可以继续使用 Studio 中以 Kinesis Data Analytics 为源使用。

使用 Studio 笔记本与 Amaze Flink 的 Kinesis 数据Analyyics

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。只需在Amazon管理控制台中单击几下，即可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。

笔记本是基于 Web 的开发环境。使用笔记本电脑，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 笔记本电脑使用由 Apache Zeppelin 提供支持的笔记本电脑，并使用 Apache Flink 作为流处理引擎。Studio 笔记本无缝结合了这些技术，使所有技能组的开发人员都能访问数据流的高级分析。

Apache Zeppelin 为您的 Studio 笔记本电脑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
- 控制输出格式以便于分析

要开始使用 Kinesis Data Analytics 和 Apache Zeppelin，请参阅创建 Studio 笔记本教程 (p. 58)。有关 Amaze Zepelin的更多信息，请参阅 A maze Zepelin文档。

在笔记本中，你可以使用 SQL、Python 或 Scala 中的 Apache Flink Table API 和 SQL 或者 Scala 中的 DataStream API 对查询进行建模。只需点击几下，您就可以将 Studio 笔记本升级为适用于生产工作负载的持续运行、非交互式 Kinesis Data Analytics 流处理应用程序。

本主题包含下列部分：

创建 Studio 笔记本电脑

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，并提供查询以分析来自数据源的数据。

您的应用程序包含下列组件：

- 数据源，例如亚马逊 MSK 集群、Kinesis 数据流或 Amazon S3 存储桶。

- Amazon Glue 数据库。该数据库包含表，用于存储您的数据源和目标架构和终端节点。有关更多信息，请参阅使用 Amazon Glue (p. 57)：
- 您的应用程序代码。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《Amaz e Flink 应用程序开发人员指南》中的以下主题：
  - 应用程序并行度和扩展：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量。如果您的查询有多个执行路径，例如在以下情况下，它们也可以利用更高的并行度：
    - 处理 Kinesis 数据流的多个分片时
    - 使用KeyBy运算符对数据进行分区时。
    - 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 中的应用程序扩展。
  - 日志记录和监控：有关应用程序日志记录和监控的信息，请参阅 Amazon Kinesis Data Analytics for Apache Flink 中的日志记录和监控。
  - 您的应用程序使用检查点和保存点来容错。默认情况下，Studio 笔记本电脑不启用检查点和保存点。

您可以使用Amazon Web Services Management Console或创建 Studio 笔记本Amazon CLI。

从控制台创建应用程序时，您可以选择以下选项：

- 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上选择实时处理数据。
- 在 Kinesis Data Analytics 控制台中，选择 Studio 选项卡，然后选择创建 Studio 笔记本。

有关如何使用Amazon Web Services Management Console或创建 Studio 笔记本的教程Amazon CLI，请参阅教程：在 Kinesis Data Analytics 中创建 Studio 笔记本 (p. 58)。

有关更高级的 Studio 笔记本电脑解决方案的示例，请参阅A mazon Kinesis Data Analytics 工作室上的 Apache Flin k。

流媒体数据的交互式分析

你使用由 Apache Zeppelin 提供支持的无服务器笔记本来与你的流媒体数据进行交互。您的笔记本可以有多个笔记，每个笔记可以包含一个或多个段落，您可以在其中编写代码。

以下 SQL 查询示例显示了如何从数据源检索数据：

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

有关 Flink Streaming SQL 查询的更多示例，请参阅示例和教程 (p. 58)下文，以及 Apache Flink 文档中的查询。

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

Flink 解释器

您可以使用解释器指定 Kinesis Data Analytics 使用哪种语言来运行应用程序。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 解释器。

如果您使用`%flink.pyflink`或`%flink.ipyflink`作为口译员，则需要使用在笔记本中`ZeppelinContext`对结果进行可视化。

有关更多 PyFlink 具体示例，请参阅使用 Kinesis Data Analytics 工作室和 Python 以交互方式查询您的数据流。

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`	`StreamExecutionEnvironment`
`benv`	`ExecutionEnvironment`
`stenv`	`StreamTableEnvironment #######`
`btenv`	`BatchTableEnvironment #######`
`stenv_2`	`StreamTableEnvironment ### flink ###`
`btenv_2`	`BatchTableEnvironment ### flink ###`

您可以使用以下变量访问 Python 表环境资源：

变量	资源
`s_env`	`StreamExecutionEnvironment`
`b_env`	`ExecutionEnvironment`
`st_env`	`StreamTableEnvironment #######`
`bt_env`	`BatchTableEnvironment #######`

变量	资源
st_env_2	StreamTableEnvironment ### flink ###
bt_env_2	BatchTableEnvironment ### flink ###

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

作为具有持久状态的应用程序部署

您可以编译代码并将其导出到 Amazon S3。您可以将您在笔记中编写的代码升级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：使用 Studio 笔记本，您可以交互式开发代码，实时查看代码结果，并在笔记中将其可视化。部署笔记以在流媒体模式下运行后，Kinesis Data Analytics 会为您创建一个持续运行、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扩展的应用程序。

Note

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中。

只有满足以下条件时，才能从 Studio 笔记本部署笔记：

- 段落必须按顺序排序。部署应用程序时，注释中的所有段落将按注释中出现的顺序 (left-to-right, top-to-bottom) 执行。您可以通过在笔记中选择 “运行所有段落” 来检查此顺序。
- 你的代码是 Python 和 SQL 或 Scala 和 SQL 的组合。我们目前不支持 Python 和 Scala deploy-as-application。
- 您的笔记应仅包含以下解释器：%flink、%flink.ssql、%flink.pyflink、%flink.ipyflink、%md。
- 不支持使用 Zeppelz in 上下文对象。除了记录警告外，什么都不返回的方法什么都不做。其他方法会引发 Python 异常或无法在 Scala 中编译。
- 注释必须生成一个 Apache Flink 作业。
- 不支持将带有动态表单的注释作为应用程序部署。
- %md (M arkdown) 段落在作为应用程序部署时将被跳过，因为这些段落应该包含人类可读的文档，不适合作为生成的应用程序的一部分运行。
- 在部署为应用程序时，将跳过因在 Zeppelin 中运行而被禁用的段落。即使禁用的段落使用了不兼容的解释器，例如，%flink.ipyflink在带有%flinkand %flink.ssql解释器的注释中，在将注释部署为应用程序时也会跳过该解释器，并且不会导致错误。
- 必须至少有一个段落包含已启用运行的源代码（Flink SQL PyFlink 或 Flink Scala），应用程序部署才能成功。
- 在通过注释部署的应用程序中，在段落（例如%flink.ssql(parallelism=32)）内的解释器指令中设置并行度将被忽略。相反，您可以通过Amazon Command Line Interface或Amazon API 更新已部署的Amazon Web Services Management Console应用程序，以根据应用程序所需的并行度级别更改 Parallelism 和/或 ParallelismPer KPU 设置，也可以为已部署的应用程序启用自动扩展。

斯卡拉/Python 标准

- 你的 Scala 或 Python 代码不能使用BatchExecutionEnvironment或BatchTableEnvironment（benv，，btenvbtenv_2对于 Scala；b_env，，bt_envbt_env_2对于 Python）。
- 在你的 Scala 或 Python 代码中，使用 Blink 规划器（senvstenv对于 Scala；s_env，用st_env于 Python），而不是较早的 “Flink” 规划器（stenv_2对于 Scala，用st_env_2于 Python）。Apache Flink 项目建议在生产用例中使用 Blink 计划器，这是 Zeppelin 和 Flink 中的默认计划器。

- 你的 Python 段落不得使用外壳调用/赋值，也不得使用！IPython 魔法命令，比如`%conda`在注释中`%timeit`或打算作为应用程序部署的注释中。
- 你不能使用 Scala 案例类作为传递给高阶数据流运算符（如`map`和）的函数的参数`filter`。有关 Scala 案例类的信息，请参阅 Scala 文档中的案例类。

SQL 条件

- 不允许使用简单的 SELECT 语句，因为没有一个等效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（`USECREATE`、`ALTER`、`DROP`、、`SET`、`RESET`）必须位于 DML (`INSERT`) 语句之前。这是因为段落中的 DML 语句必须作为单个 Flink 作业一起提交。
- 最多应该有一个段落中包含 DML 语句。这是因为，对于该 deploy-as-application 功能，我们只支持向 Flink 提交单个作业。

有关更多信息和示例，请参阅使用带有 Amazon Kinesis 数据分析、Amazon Translate 和 Amazon Comprehend 的 SQL 函数Amazon Translate、编辑和分析流媒体数据。

Studio 笔记本的 IAM 权限

当您通过创建 Studio 笔记本时，Kinesis Data Analytics 会为您创建 IAM 角色Amazon Web Services Management Console。它还与该角色关联了一个允许以下访问的策略：

服务	访问
CloudWatch 日志	List
Amazon EC2	List
Amazon Glue	读取、写入
Kinesis Data Analytics	Read
Kinesis Data Analytics V2	Read
Amazon S3	读取、写入

连接器和依赖关系

连接器使您能够通过各种技术读取和写入数据。Kinesis Data Analytics 将三个默认连接器与您的 Studio 笔记本电脑捆绑在一起。您也可以使用自定义连接器。有关连接器的更多信息，请参阅 Apache Flink 文档中的表和 SQL 连接器。

默认连接器

如果您使用创建 Studio 笔记本，则默认情况下，Kinesis Data Analytics 包含以下自定义连接器：`flink-sql-connector-flink`、`flink-connector-kafka_2.12`和`aws-msk-iam-auth`。Amazon Web Services Management Console要在不使用这些自定义连接器的情况下通过控制台创建 Studio 笔记本，请选择“使用自定义设置创建”选项。然后，当你进入配置页面时，清除两个连接器旁边的复选框。

如果您使用 CreateApplicationAPI 创建 Studio 笔记本，则默认情况下不包括`flink-sql-connector-flink`和`flink-connector-kafka`连接器。要添加它们，请在`CustomArtifactsConfiguration`数据类型`MavenRefernce`中将它们指定为 a，如以下示例所示。

`aws-msk-iam-auth`连接器是与 Amazon MSK 一起使用的连接器，它包含自动向 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": "1.15.2"
   }
}]

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": "1.15.2"
   }
}]
```

要将这些连接器添加到现有笔记本中，请使用 UpdateApplicationAPI 操作并在`CustomArtifactsConfigurationUpdate`数据类型`MavenReference`中将它们指定为 a。

Note

您可以`failOnError`将表 API 中的`flink-sql-connector-kinesis`连接器设置为 true。

依赖关系和自定义连接器

要使用向 Studio 笔记本添加依赖项或自定义连接器，请按照以下步骤操作：Amazon Web Services Management Console

1. 将自定义连接器的文件上载到 Amazon S3。
2. 在中Amazon Web Services Management Console，选择 “自定义创建” 选项来创建 Studio 笔记本。
3. 按照 Studio 笔记本创建工作流程进行操作，直到进入 “配置” 步骤。
4. 在自定义连接器部分中，选择添加自定义连接器。
5. 指定依赖关系或自定义连接器的 Amazon S3 位置。
6. 选择保存更改。

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

Note

在包含依赖项或自定义连接器时，还必须包含其所有未捆绑在其中的传递依赖关系。

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无法在查询中以其他方式表达的自定义逻辑。你可以使用 Python 或 Java 或 Scala 等 JVM 语言在 Studio 笔记本的段落中实现 UDF。您还可以将包含以 JVM 语言实现的 UDF 的外部 JAR 文件添加到您的 Studio 笔记本中。

在实现注册子类抽象类`UserDefinedFunction`（或你自己的抽象类）的 JAR 时，在 Apache Maven 中使用提供的作用域，在 Gradle 中使用`compileOnly`依赖关系声明，在 SBT 中使用提供的作用域，或者在 UDF 项目构建配置中使用等效指令。这允许 UDF 源代码根据 Flink API 进行编译，但是 Flink API 类本身并不包含在构建构件中。请参考 UDF jar 示例中的这个 pom，它符合 Maven 项目的此类先决条件。

Note

有关设置示例，请参阅：M achine Learnin g 博客中的使用 Amazon Translatics、Amazon Translate 和 Amazon Comprehend 的 SQL 函数翻译、编辑和分析流数据。Amazon

要使用控制台将 UDF JAR 文件添加到 Studio 笔记本，请按照以下步骤操作：

1. 将您的 UDF JAR 文件上载到 Amazon S3。
2. 在中Amazon Web Services Management Console，选择 “自定义创建” 选项来创建 Studio 笔记本。
3. 按照 Studio 笔记本创建工作流程进行操作，直到进入 “配置” 步骤。
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 文件或实现 UDF 的 ZIP 文件的 Amazon S3 位置。
6. 选择保存更改。

要在使用 CreateApplicationAPI 创建新的 Studio 笔记本时添加 UDF JAR，请在`CustomArtifactConfiguration`数据类型中指定 JAR 位置。要将 UDF JAR 添加到现有 Studio 笔记本中，请调用 UpdateApplicationAPI 操作并在`CustomArtifactsConfigurationUpdate`数据类型中指定 JAR 位置。或者，您可以使用将 UDF JAR 文件添加Amazon Web Services Management Console到 Studio 笔记本中。

有关用户定义函数的注意事项

- Kinesis Data Analytics Studio 使用 Apache Zeppelin 术语，其中笔记本是可以包含多个音符的齐柏林飞艇实例。然后，每个注释可以包含多个段落。在 Kinesis Data Analytics Studio 中，解释器过程可以在笔记本的所有笔记中共享。因此，如果您在一个笔记中使用函数进行显式createTemporarySystem函数注册，则可以在同一笔记本的另一个笔记中按原样引用相同的函数。

  但是，“以应用程序身份部署” 操作适用于单个笔记，而不是笔记本中的所有笔记。当您以应用程序身份执行部署时，仅使用活动笔记的内容来生成应用程序。在其他笔记本电脑中执行的任何显式函数注册都不是生成的应用程序依赖项的一部分。此外，在 “部署为应用程序” 选项期间，通过将 JAR 的主类名转换为小写字符串来进行隐式函数注册。

  例如，如果`TextAnalyticsUDF`是 UDF JAR 的主类，则隐式注册将生成函数名`textanalyticsudf`。因此，如果 Studio 注释 1 中的显式函数注册如下所示，那么`myNewFuncNameForClass`由于共享解释器，该笔记本中的所有其他笔记（比如注释 2）都可以按名称引用该函数：

```
stenv.createTemporarySystemFunction("myNewFuncNameForClass", new
TextAnalyticsUDF())
```

但是，在注释 2 上以应用程序身份部署操作期间，这种显式注册将不包含在依赖项中，因此已部署的应用程序将无法按预期运行。由于隐式注册，默认情况下，对该函数的所有引用都应为withtextanalyticsudf 和 notmyNewFuncNameForClass。

如果需要注册自定义函数名，则 note 2 本身应包含另一段用于执行另一次显式注册的段落，如下所示：

```
%flink(parallelism=l)
import com.amazonaws.kinesis.udf.textanalytics.TextAnalyticsUDF
# re-register the JAR for UDF with custom name
stenv.createTemporarySystemFunction("myNewFuncNameForClass", new TextAnalyticsUDF())
```

```
%flink. ssql(type=update, parallelism=1)
INSERT INTO
    table2
SELECT
    myNewFuncNameForClass(column_name)
FROM
    table1
;
```

- 如果您的 UDF JAR 包含 Flink SDK，请配置您的 Java 项目，以便 UDF 源代码可以针对 Flink SDK 进行编译，但是 Flink SDK 类本身并不包含在构建工件中，例如 JAR。

  你可以在 Apache Maven 中使用provided作用域，在 Gradle 中使用compileOnly依赖关系声明，在 SBT 中使用provided作用域，或者在其 UDF 项目构建配置中使用等效指令。你可以从 UDF jar 示例中引用这个 pom，它符合 maven 项目的此类先决条件。有关完整 step-by-step 教程，请参阅此Machine Le Amazonarning 博客。

启用检查点检验工具

您可以使用环境设置启用检查点。有关检查点的信息，请参阅《Kinesis Data Analytics 开发人员指南》中的容错功能。

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（默认值）：

```
// set mode to exactly-once (this is the default)
```

```
stenv.getCheckpointConfig.setCheckpointingMode(CheckpointingMode.EXACTLY_ONCE)
```

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

```
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，需要指定包含连接信息Amazon Glue的数据库。访问数据源和接收器时，需要指定数据库中包含的Amazon Glue表。您的Amazon Glue表提供对定义数据源和目标的位置、架构和参数的Amazon Glue连接的访问权限。

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. 57)：

有关如何设置Amazon Glue连接、数据库和表以用于 Studio 笔记本电脑的示例，请参阅创建 Studio 笔记本教程 (p. 58)教程创建Amazon Glue数据库 (p. 59)中的。

表属性

除数据字段外，您的Amazon Glue表还使用表格属性向 Studio 笔记本提供其他信息。Kinesis Data Analytics使用以下Amazon Glue表格属性：

- 使用 Apache Flink 时间值 (p. 57)：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. 58)：这些属性提供有关您的数据流的信息。

要向Amazon Glue表中添加属性，请执行以下操作：

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. 从表列表中，选择应用程序用来存储其数据连接信息的表。选择操作，编辑表格详细信息。
3. 在“表属性”下，输入**kinesisanalytics.proctime** fo r 键和**user_action_time** v alue。

使用 Apache Flink 时间值

Apache Flink 提供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表上定义属性，告诉 Kinesis Data Analytics 运行时将这些值发送到指定字段中。

您在表属性中使用的键和值如下所示：

时间戳类型	键	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	键	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.warker ## .milseconds	The watermark interval in milliseconds

使用 Flink 连接器和格式属性

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

连接器类型	键	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. `#####` or `timestamp`.
Kinesis	`format`	The format used to deserialize and serialize Kinesis data stream records, e.g. `json` or `csv`.
	`aws. ##`	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，请参阅您的连接器文档。

示例和教程

主题

- 教程：在 Kinesis Data Analytics 中创建 Studio 笔记本 (p. 58)
- 教程：作为具有持久状态的应用程序部署 (p. 71)
- 示例 (p. 73)

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

确保您的版本Amazon CLI为 2 或更高版本。要安装最新版本Amazon CLI，请参阅安装、更新和卸载Amazon CLI版本 2。

创建Amazon Glue数据库

您的 Studio 笔记本使用Amazon Glue数据库存储有关您的亚马逊 MSK 数据源的元数据。

创建Amazon Glue数据库

1. 打开 Amazon Glue 控制台，地址：https://console.aws.amazon.com/glue/。
2. 选择 Add database（添加数据库）。在添加数据库窗口中**default**，输入数据库名称。选择创建。

后续步骤

通过本教程，你可以创建使用 Kinesis Data Streams 或亚马逊 MSK 的 Studio 笔记本：

- Kinesis Data Streams (p. 59)：使用 Kinesis Data Streams，您可以快速创建使用 Kinesis 数据流作为源。您只需要创建 Kinesis 数据流作为依赖资源。
- Amazon MSK (p. 63)：使用 Amazon MSK，您可以创建一个使用 Amazon MSK 集群作为源的应用程序。您需要创建一个 Amazon VPC、一个 Amazon EC2 客户端实例和一个 Amazon MSK 集群作为依赖资源。

使用 Kinesis Data Streams 创建创建 Guo

本教程描述如何创建使用 Kinesis 数据流作为源。

本教程包含以下部分：

设置

在创建 Studio 笔记本之前，创建 Kinesis 数据流 (`ExampleInputStream`)。您的应用程序使用此流作为应用程序源。

您可以使用 Amazon Kinesis 控制台或以下Amazon CLI命令创建此直播。有关控制台的说明，请参阅Amazon Kinesis 数据流开发者指南中的创建和更新数据流。命名该流**`ExampleInputStream`**并将打开的分片数量设置为**1**。

要使用创建直播 (ExampleInputStream)Amazon CLI，请使用以下 Amazon Kinesiscreate-streamAmazon 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

您的 Studio 笔记本使用Amazon Glue数据库存储有关 Kinesis Data Streams 数据源的元数据。

> 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，您可以按照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，您可以签入Amazon Glue以确保正确创建了此表。

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数据库，请从左侧导航栏中选择数据库。选择 “添加数据库”。在添加数据库窗口中**default**，输入数据库名称。选择创建。
3. 在左侧导航栏中，选择表。在表格页面中，选择添加表，手动添加表。
4. 在设置表的属性页面中**stock**，输入表名。确保选择之前创建的数据库。选择下一步。
5. 在添加数据存储页面中，选择 Kinesis。对于直播名称，输入**ExampleInputStream**。对于 Kinesis 源网址，选择回车键**https://kinesis.us-east-1.amazonaws.com**。如果您复制并粘贴 Kinesis 源网址，请务必删除所有开头或结尾的空格。选择下一步。
6. 在分类页面中，选择 JSON。选择下一步。
7. 在 “定义架构” 页面中，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**

   选择下一步。
8. 在下一页上，验证您的设置，然后选择完成。
9. 从表列表中选择新创建的表。
10. 选择编辑表并添加带有键kinesisanalytics.proctime和值的属性proctime。
11. 选择 Apply（应用）。

使用 Kinesis Data Streams 创建创建 Dio

现在，您已经创建了应用程序使用的资源，接下来就可以创建您的 Studio 笔记本了。

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

- 使用创建 Studio 笔记本Amazon Web Services Management Console (p. 61)
- 使用创建 Studio 笔记本Amazon CLI (p. 61)

使用创建 Studio 笔记本Amazon Web Services Management Console

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics/home?region=us-east-1#/applications/dashboard。
2. 在 Kinesis Data Analytics 应用程序页面中，选择 Studio 选项卡。选择创建 Studio 笔记本。

   Note

   您还可以从亚马逊 MSK 或 Kinesis Data Streams 控制台创建 Studio 笔记本，方法是选择您的输入亚马逊 MSK 集群或 Kinesis 数据流，然后选择 “实时处理数据”。
3. 在创建创建 Studio 笔记本页面上，提供以下信息

   - 输入笔记本**MyNotebook**的名称。
   - 为 AmazonGlue 数据库选择默认值。

   选择创建 Studio 笔记本。
4. 在MyNotebook页面中，选择 “运行”。等待状态显示正在运行。笔记本电脑运行时收费。

使用创建 Studio 笔记本Amazon CLI

要使用创建 Studio 笔记本Amazon CLI，请执行以下操作：

1. 验证账户 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`并将以下权限添加到控制台自动创建的角色中。

   ```
   "kinesis:GetShardIterator",

   "kinesis:GetRecords",

   "kinesis:ListShards"
   ```
3. 创建以下内容的名为 `create.json` 的文件。用您的信息替换占位符值。

   ```
   {
       "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. 运行以下命令创建应用程序。

   ```
   aws kinesisanalyticsv2 create-application --cli-input-json file://create.json
   ```
5. 命令完成后，您将看到显示新 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\
```

将数据发送到您的 Kinesis 数据流

要将测试数据发送到您的 Kinesis 数据流，请执行以下操作：

1. 打开 Kinesis 数据生成器。
2. 选择“使用”创建 Cognito 用户 CloudFormation。
3. Amazon CloudFormation控制台以 Kinesis 数据生成器模板打开。选择下一步。
4. 在指定堆栈详细信息页面中，输入您的 Cognito 用户的用户名和密码。选择下一步。
5. 在配置堆栈选项页面中，选择下一步。
6. 在“查看 Kinesis-Data-Generator-Cognito-User”页面中，选择Amazon CloudFormation 可能创建 IAM 资源的“我确认”。 复选框。选择Create Stack（创建堆栈）。
7. 等待Amazon CloudFormation堆栈完成创建。堆栈完成后，在Amazon CloudFormation控制台中打开 Kinesis-Data-Generator-Cognito-User 堆栈，然后选择“输出”选项卡。打开列出的KinesisDataGeneratorUrl输出值的 URL。
8. 在 Amazon Kinesis 数据生成器页面中，使用您在步骤 4 中创建的证书登录。
9. 在下一页上，提供以下值

区域	**us-east-1**
传输流/传输流	**ExampleInputStream**
每秒记录数	**1**

对于“记录模板”，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. 生成器会将数据发送到 Kinesis 数据流。

在完成下一节的同时，让生成器保持运行状态。

测试您的工作室笔记本

在本节中，您将使用 Studio 笔记本来查询 Kinesis 数据流中的数据。

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics/home?region=us-east-1#/applications/dashboard。
2. 在 Kinesis Data Analytics 应用程序页面上，选择 Studio 笔记本选项卡。选择MyNotebook。
3. 在MyNotebook页面中，选择 “在 Apache Zeppelin 中打开”。

   Apache Zeppelin 接口在新选项卡中打开。
4. 在《欢迎来到齐柏林》中！ 页面，选择 “齐柏林笔记”。
5. 在 Zeppelin Note 页面中，在新笔记中输入以下查询：

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

   选择运行图标。

   片刻之后，笔记会显示来自 Kinesis 数据流的数据。

要打开 Apache Flink 控制面板供应用程序查看操作方面，请选择 FLINK JOB。有关 Flink 仪表板的更多信息，请参阅 Kinesis D ata Analytics 开发者指南中的 Apache Flin k 仪表板。

有关 Flink 串流 SQL 查询的更多示例，请参阅 Apache Flink 文档中的查询。

使用亚马逊 MSK 创建 Studio 笔记本

本教程描述如何创建使用 Amazon MSK 集群作为源的 Studio 笔记本。

本教程包含以下部分：

设置

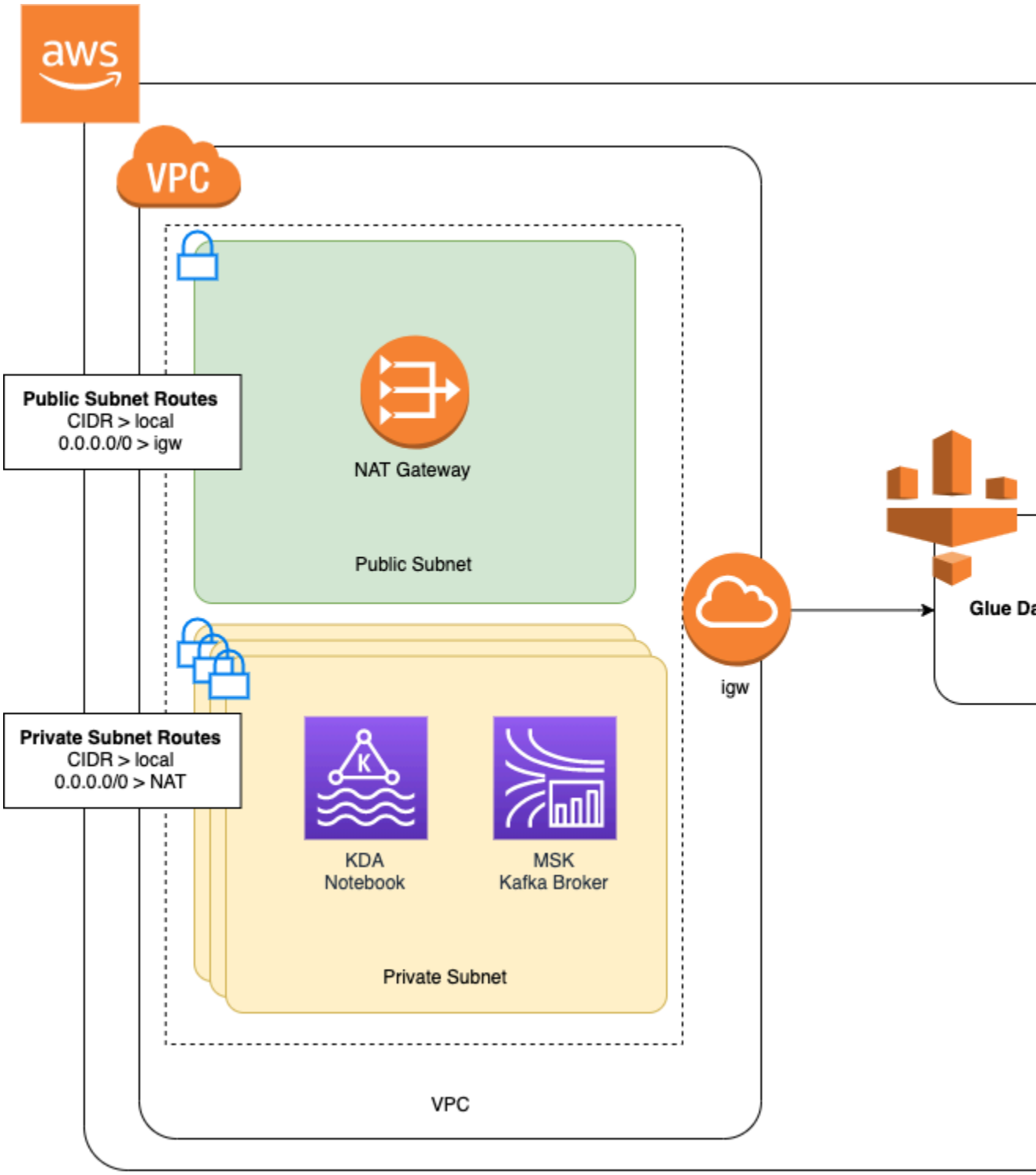
在此教程中，您需要一个允许明文访问的 Amazon MSK 集群。如果您尚未设置 Amazon MSK 集群，请按照使用 Amazon MSK 入门教程创建 Amazon VPC、亚马逊 MSK 集群、主题和 Amazon EC2 客户端实例。

在学习教程时，执行以下操作：

- 在步骤 3：创建 Amazon MSK 集群中，在步骤 4 中，将`ClientBroker`值从更改`TLS`为**`PLAINTEXT`**。

将 NAT 网关添加到您的 VPC

如果您按照使用 Amazon MSK 入门教程创建了 Amazon MSK 集群，或者您的现有 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。



要为您的亚马逊 VPC 创建 NAT 网关，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. 从左侧导航栏中选择 NAT 网关。
3. 在 NAT 网关页面上，选择创建 NAT 网关。

4. 在创建 NAT 网关页面上，提供以下值：

名称-可选	**`ZeppelinGateway`**
子网	AmazonKafkaTutorialSubnet1
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 Create NAT Gateway（创建 NAT 网关）。

5. 在左侧导航栏上，选择 Roty Table（路由表）。
6. 选择 Create Route Table。
7. 在创建路由表页面上，提供以下信息：

   - 名称标签：**`ZeppelinRouteTable`**
   - VPC：选择您的 VPC（例如 AmazonKafkaTutorialVPC）。

   选择创建。
8. 在路由表列表中，选择ZeppelinRouteTable。选择路线选项卡，然后选择编辑路线。
9. 在编辑路径页面中，选择添加路线。
10. 在 “目标位置” 中，输入**`0.0.0.0/0`**。对于目标，选择 NAT 网关，ZeppelinGateway。选择 “保存路线”。选择 Close（关闭）。
11. 在路由表页面上，ZeppelinRouteTable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中，选择 AmazonKafkaTutorialSubnet2 和 AmazonKafkaTutorialSubnet3。选择保存。

创建Amazon Glue连接和表

您的 Studio 笔记本使用Amazon Glue数据库存储有关您的亚马逊 MSK 数据源的元数据。在本节中，您将创建一个描述如何访问您的 Amazon MSK 集群的Amazon Glue连接，以及一个描述如何向客户端（例如 Studio 笔记本）呈现数据源中的数据的Amazon Glue表。

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，请从左侧导航栏中选择数据库。选择 “添加数据库”。在添加数据库窗口中**`default`**，输入数据库名称。选择创建。
3. 从左侧导航栏中选择 “连接”。选择 “添加连接”。
4. 在 “添加连接” 窗口中，提供以下值：

   - 在连接名称中，输入**`ZeppelinConnection`**。
   - 对于 Connection type (连接类型)，选择 Kafka。
   - 对于 Kafka 引导服务器 URL，请提供集群的引导代理字符串。您可以从 MSK 控制台获取引导代理，也可以通过输入以下 CLI 命令来获取：

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

   - 取消选中 “需要 SSL 连接” 复选框。

选择下一步。

5. 在 VPC 页面中，提供以下值：

   - 对于 VPC，选择您的 VPC 的名称（例如 AmazonKafkaTutorialVPC。）
   - 对于子网，选择 AmazonKafkaTutorialSubnet2。
   - 对于安全组，选择所有可用组。

   选择下一步。

6. 在 “连接属性”/“连接访问权限” 页面中，选择 “完成”。

创建表

Note

您可以按照以下步骤中的说明手动创建表，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，您可以签入Amazon Glue以确保正确创建了此表。

1. 在左侧导航栏中，选择 Table（表）。在表格页面中，选择添加表，手动添加表。
2. 在设置表的属性页面中**`stock`**，输入表名。确保选择之前创建的数据库。选择下一步。
3. 在添加数据存储页面中，选择 Kafka。在主题名称中，输入您的主题名称（例如 AmazonKafkaTutorialTopic）。对于 “连接”，选择ZeppelinConnection。
4. 在分类页面中，选择 JSON。选择下一步。
5. 在 “定义架构” 页面中，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`**

选择下一步。

6. 在下一页上，验证您的设置，然后选择 Finish（完成）。
7. 从表列表中选择新创建的表。
8. 选择编辑表并添加带有键`kinesisanalytics.proctime`和值的属性`proctime`。
9. 选择 Apply（应用）。

使用亚马逊 MSK 创建 Studio 笔记本

现在，您已经创建了应用程序使用的资源，接下来就可以创建您的 Studio 笔记本了。

您可以使用Amazon Web Services Management Console或创建应用程序Amazon CLI。

- 使用创建 Studio 笔记本Amazon Web Services Management Console (p. 67)
- 使用创建 Studio 笔记本Amazon CLI (p. 61)

Note

您也可以从 Amazon MSK 控制台创建 Studio 笔记本，方法是选择现有集群，然后选择 “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，网址为 https://console.aws.amazon.com/kinesisanalytics/home?region=us-east-1#/applications/dashboard。
2. 在 Kinesis Data Analytics 应用程序页面中，选择 Studio 选项卡。选择创建 Studio 笔记本。

   Note

   要从亚马逊 MSK 或 Kinesis Data Streams 控制台创建 Studio 笔记本，请选择您的输入亚马逊 MSK 集群或 Kinesis 数据流，然后选择实时处理数据。
3. 在创建 Studio 笔记本页面中，提供以下信息：

   - 输入 St**MyNotebook** udio 笔记本名称。
   - 为 AmazonGlue 数据库选择默认值。

   选择创建 Studio 笔记本。
4. 在MyNotebook页面中，选择 “配置” 选项卡。在 “网络” 部分中，选择 “编辑”。
5. 在编辑网络 MyNotebook页面中，选择基于 Amazon MSK 集群的 VPC 配置。为亚马逊 MSK 集群选择您的亚马逊 MSK 集群。选择保存更改。
6. 在MyNotebook页面中，选择 “运行”。等待状态显示正在运行。

使用创建 Studio 笔记本Amazon CLI

要使用创建 Studio 笔记本Amazon CLI，请执行以下操作：

1. 确认您具有以下信息。您需要这些值来创建应用程序。

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

```
{
    "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应用程序：

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

4. 在运行命令后，您应该会看到类似于以下内容的输出，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应用程序。将样本值替换为您的账户 ID。

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

向您的亚马逊 MSK 集群发送数据

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

1. Connect 您的 Amazon EC2 客户端。
2. 运行以下命令安装 Python 版本 3、Pip 和 Kafka for 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. 通过输入以下命令，在客户端计算机Amazon CLI上配置：

```
aws configure
```

提供您的账户凭证，**us-east-1**以及region.

4. 创建以下内容的名为 stock.py 的文件。将示例值替换为您的 Amazon MSK 集群的 Bootstrap Brokers 字符串，如果您的主题不是，则更新主题名称 AmazonKafkaTutorialTopic：

```
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. 在完成以下部分的同时，让脚本保持运行状态。

测试您的工作室笔记本

在本节中，您将使用 Studio 笔记本从您的亚马逊 MSK 集群查询数据。

1. 打开 Kinesis Data Analytics 控制台，网址为 https://console.aws.amazon.com/kinesisanalytics/home?region=us-east-1#/applications/dashboard。
2. 在 Kinesis Data Analytics 应用程序页面上，选择 Studio 笔记本选项卡。选择MyNotebook。
3. 在MyNotebook页面中，选择 “在 Apache Zeppelin 中打开”。

   Apace Zepelin 界面将在新的选项卡中打开。
4. 在《欢迎来到齐柏林》中！ 页面，选择 “齐柏林飞艇新笔记”。
5. 在 Zeppelin Note 页面中，在新笔记中输入以下查询：

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

   选择运行图标。

   该应用程序会显示来自于 Amazon MSK 集群的数据。

要打开 Apache Flink 控制面板供应用程序查看操作方面，请选择 FLINK JOB。有关 Flink 仪表板的更多信息，请参阅 Kinesis D ata Analytics 开发者指南中的 Apache Flin k 仪表板。

有关 Flink 串流 SQL 查询的更多示例，请参阅 Apache Flink 文档中的查询。

清理您的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，然后选择删除。

删除您的Amazon Glue数据库和连接

1. 打开 Amazon Glue 控制台，地址：https://console.aws.amazon.com/glue/。
2. 从左侧导航栏中选择数据库。选中 “默认” 旁边的复选框将其选中。选择操作，删除数据库。确认您的选择。
3. 从左侧导航栏中选择 “连接”。选中旁边的复选框ZeppelinConnection将其选中。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. 从左侧导航栏中选择 “角色”。
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

当您使用控制台创建应用程序时，控制台会为您创建 CloudWatch 日志组和日志流。如果您使用创建了应用程序，则没有日志组和流媒体Amazon CLI。

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
2. 从左侧导航栏中选择日志组。
3. 选择 /aws/kinesis-analytics/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。

清除资源 Kinesis Data Streams 资源

要删除你的 Kinesis 直播，请打开 Kinesis Data Streams 控制台，选择你的 Kinesis 直播，然后选择操作、删除。

清除资源 MSK 资源

如果您为本教程创建了 Amazon MSK 集群，请执行本部分中的步骤。本节介绍如何清理您的Amazon EC2客户端实例、亚马逊 VPC 和亚马逊 MSK 集群。

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. 通过 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/ 打开亚马逊 MSK 控制台。
2. 选择AmazonKafkaTutorialCluster。选择删除。**`delete`**在出现的窗口中输入，然后确认您的选择。

终止实例

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. 从左侧导航栏中选择实例。
3. 选中旁边的复选框ZeppelinClient将其选中。
4. 选择实例状态，终止实例。

删除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. 从左侧导航栏中选择网络接口。
3. 在搜索栏中输入您的 VPC ID，然后按回车键进行搜索。
4. 选中表格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。在出现的窗口中，在“强制分离”下选择“启用”。选择 Detach，然后等待所有网络接口达到“可用”状态。
6. 选中表格标题中的复选框可再次选择所有显示的网络接口。
7. 依次选择操作和删除。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. 选择 AmazonKafkaTutorialVPC。选择操作，删除 VPC。输入**delete**并确认删除。

教程：作为具有持久状态的应用程序部署

The following tutorial demonstrates how to deploy a Studio notebook as a Kinesis Data Analytics application with durable state.

本教程包含以下部分：

- 设置 (p. 71)
- 使用持久状态部署应用程序Amazon Web Services Management Console (p. 71)
- 使用持久状态部署应用程序Amazon CLI (p. 72)

设置

按照 Kinesis Data Streams 或亚马逊 MSK 创建新的 Studio 笔记本。创建 Studio 笔记本教程 (p. 58)为 Studio 笔记本命名`ExampleTestDeploy`。

使用持久状态部署应用程序Amazon Web Services Management Console

1. 在控制台中的应用程序代码位置（可选）下添加要存储打包代码的 S3 存储桶位置。这使这些步骤能够直接从笔记本部署和运行您的应用程序。
2. 向应用程序角色添加所需权限，以启用您用来读写 Amazon S3 存储桶以及启动 Kinesis Data Analytics 应用程序的角色：

    - 亚马逊 S3FullAccess

- AmazonKinesisAnalyticsFullAccess
- 访问您的来源、目的地和 VPC（如果适用）。有关更多信息，请参阅Studio 笔记本的 IAM 权限 (p. 53)：

3. Use the following sample code:

```
%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，您将在笔记本中每个笔记的右上角看到一个新的下拉列表，上面有笔记本的名称。您可执行以下操作：

- 在中查看 Studio 笔记本电脑设置Amazon Web Services Management Console。
- Build your Zeppelin Note and export it to Amazon S3. 此时，为您的应用程序提供一个名称并选择 “生成并导出”。导出完成后，您将收到通知。
- 如果需要，您可以在 Amazon S3 中的可执行文件上查看和运行任何其他测试。
- 构建完成后，您将能够将代码部署为具有持久状态和自动缩放的 Kinesis 流媒体应用程序。
- 使用下拉列表并选择 “将 Zeppelin Note 部署为 Kinesis 直播应用程序”。查看应用程序名称并选择通过Amazon控制台部署。
- Amazon Web Services Management Console 请注意，应用程序名称、并行度、代码位置、默认 Glue DB、VPC（如果适用）和 IAM 角色已预先填充。验证 IAM 角色是否具有访问您的来源和目标所需的权限。默认情况下，快照处于启用状态，用于持久的应用程序状态管理。
- 选择创建应用程序。
- 您可以选择配置和修改任何设置，然后选择 Run 启动您的流媒体应用程序。

使用持久状态部署应用程序Amazon CLI

要使用部署应用程序Amazon CLI，您必须更新您的Amazon CLI以使用随您的 Beta 2 信息一起提供的服务模型。有关如何使用更新的服务模型设置 (p. 59)的信息，

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分析数据。

- 使用亚马逊 msk/Apache Kafka 创建表 (p. 74)
- 使用 Kinesis 创建表格 (p. 75)
- 翻滚的窗户 (p. 75)
- 滑动式窗口 (p. 75)
- 交互式SQ (p. 75)
- BlackHole SQL 连接器 (p. 76)
- 数据生成器 (p. 77)
- 交互式Scala (p. 77)
- 交互式Py (p. 78)
- 交互式 Python、SQL 和 Scala (p. 79)

- 跨账户 Kinesis 数据流 (p. 81)

有关 Apache Flink SQL 查询设置的信息，请参阅 Zeppelin 笔记本上的 Flink 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，请在应用程序的 Zeppelin Note 页面中选择 FLINK JOB。

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

有关 Apache Flink Streaming SQL 查询的更多示例，请参阅 Apache Flink 文档中的查询。

使用亚马逊 msk/Apache Kafka 创建表

您可以将亚马逊 MSK Flink 连接器与 Kinesis Data Analytics Studio 配合使用，通过纯文本、SSL 或 IAM 身份验证对您的连接进行身份验证。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'
);
```

你可以在 Apache Kafka SQL Connec tor 上将它们与其他属性结合起来。

使用 Kinesis 创建表格

在以下示例中，使用 Kinesis 创建表：

```
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 SQL 连接器。

翻滚的窗户

以下 Flink Streaming SQL 查询从ZeppelinTopic表中选择每个五秒钟的下跌窗口中的最高价格：

```
%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)
```

滑动式窗口

以下 Apache Flink Streaming SQL 查询从ZeppelinTopic表中选择每个五秒滑动窗口中的最高价格：

```
%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SQ

此示例打印事件时间和处理时间的最大值以及键值表中的值总和。确保您有the section called “数据生成器” (p. 77)正在运行的示例数据生成脚本。要在 Studio 笔记本中尝试其他 SQL 查询，例如筛选和加入，请参阅 Apache Flink 文档：Apache Flink 文档中的查询。

```
%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 SQL 连接器

S BlackHole QL 连接器不需要您创建 Kinesis 数据流或 Amazon MSK 集群来测试您的查询。有关 BlackHole SQL 连接器的信息，请参阅 Apache Flink 文档BlackHole 中的 SQL 连接器。在此示例中，默认目录是内存目录。

```
%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

此示例使用 Scala 生成示例数据。您可以使用此示例数据来测试各种查询。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Scala

这是 Scala 的翻译the section called “交互式SQ” (p. 75)。如需了解更多 Scala 示例，请参阅 Apache Flink 文档中的表 API。

```
%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Py

这是 Python 对的翻译the section called “交互式SQ” (p. 75)。要了解更多 Python 示例，请参阅 Apache Flink 文档中的表 API。

```
%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。在计划作为具有持久状态的应用程序部署的 Studio 笔记本中，可以组合使用 SQL 和 Scala。此示例向您显示了被忽略的部分以及在应用程序中以持久状态部署的部分。

```
%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其他账户中的 Kinesis 数据流，请在运行 Studio 笔记本的账户中创建服务执行角色，并在拥有该数据流的账户中创建角色信任策略。`aws.credentials.role.sessionName`在创建表 DDL 语句的 Kinesis 连接器中使用`aws.credentials.provider``aws.credentials.role.arn`、和，根据数据流创建表。

为 Studio 笔记本帐户使用以下服务执行角色。

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

对数据流账户使用`AmazonKinesisFullAccess`策略和以下角色信任策略。

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

使用以下段落作为创建表语句。

```
%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 Studio 笔记本和 Kinesis Data Analytics

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

- 在多个 Kinesis 数据流之间或在 Kinesis 数据流和 Amazon MSK 主题之间合并流数据
- 对数据流中转换后的数据进行实时可视化

- 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

要停止停留在临时状态的应用程序，请在`Force`参数设置为的情况下调用StopApplication操作`true`。有关更多信息，请参阅《Kinesis Data Analytics 开发人员指南》中的 “运行应用程序”。

取消作业

本节向你展示如何取消 Apache Flink 任务，这些任务是你无法从 Apache Zeppelin 获得的。如果你想取消这样的作业，请前往 Apache Flink 控制面板，复制任务 ID，然后在以下示例中使用它。

要取消单个任务，请执行以下操作：

```
%flink.pyflink
import requests

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

要取消所有正在运行的任务，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)
```

重启 Apache Flink 解释器

在你的 Studio 笔记本中重新启动 Apache Flink 解释器

1. 选择屏幕右上角附近的配置。
2. 选择 “解释器”。

3. 选择 “重新启动”，然后选择 “确定”。

附录：创建自定义 IAM 策略

您通常使用托管 IAM 策略来允许您的应用程序访问依赖资源。如果您需要更好地控制应用程序的权限，则可以使用自定义 IAM 策略。本节包含自定义 IAM 策略的示例。

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

以下示例策略授予访问Amazon Glue数据库的权限。

```
{
    "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

以下策略授予访问 CloudWatch 日志的权限：

```
{
      "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，则控制台会向您的应用程序角色添加访问 CloudWatch 日志所需的策略。

Kinesis Streams

您的应用程序可以使用 Kinesis Stream 作为源或目标。您的应用程序需要读取权限才能读取源流，需要写入权限才能写入目标流。

以下策略授予读取用作源的 Kinesis Stream 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/*"
    }
  ]
}
```

以下策略授予写入用作目标的 Kinesis Stream 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，则必须授予额外权限才能访问该流和该流的加密密钥。

以下策略授予访问加密源流和该流的加密密钥的权限：

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

以下策略授予访问加密目标流和该直播的加密密钥的权限：

```
{
      "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。

Amazon Kinesis for Apache Flink (DataStream API)

本节向你介绍适用于 Apache Flink 和 DataStream API 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

适用于 Flink 应用程序的 Kinesis Data Analytics 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 运行时处理输入并生成输出。

Kinesis Data Analytics 应用程序包含以下组件：

- 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：应用程序通过源 使用数据。源连接器从 Kinesis 数据流、Amazon S3 存储桶等读取数据。有关更多信息，请参阅源 (p. 14)：
- 操作符：应用程序使用一个或多个操作符 以处理数据。操作符可以转换、丰富或聚合数据。有关更多信息，请参阅DataStream API 操作员 (p. 18)：
- 接收器：应用程序使用接收器 将生成的数据发送到外部源。Sink 连接器会将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. 15)：

创建、编译和打包应用程序代码后，您可以将代码包上载到 Amazon Simple Storage Simple Simple Simple Simple Simple Si 然后，您创建 Kinesis Data Analytics 您传入代码包位置、作为流数据源的 Kinesis 数据流，通常传入接收应用程序处理过的数据流或文件位置。

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，请转到步骤 1：设置 Amazon 账户并创建管理员用户 (p. 88)。

步骤 1：设置 Amazon 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：

注册一个 Amazon Web Services 账户

如果您还没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

注册 Amazon Web Services 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。

   当您注册 Amazon Web Services 账户 时，系统将会创建一个 Amazon Web Services 账户根用户。根用户有权访问该账户中的所有 Amazon Web Services和资源。作为安全最佳实践，请 为管理用户分配管理访问权限，并且只使用根用户执行 需要根用户访问权限的任务。

Amazon注册过程完成后，会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择 My Account (我的账户) 来查看当前的账户活动并管理您的账户。

保护 IAM 用户

注册 Amazon Web Services 账户 后，启用多重身份验证（MFA）保护您的管理用户。有关说明，请参阅 IAM 用户指南中的为 为 IAM 用户（控制台）启用虚拟 MFA 设备。

要授予其他用户访问您的 Amazon Web Services 账户 资源的权限，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA 并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅 IAM 用户指南中的以下主题：

- 在您的 Amazon Web Services 账户 中创建 IAM 用户
- 适用于 Amazon 资源的访问管理
- IAM 基于身份的策略示例

授权以编程方式访问

如果用户需要在 Amazon Web Services Management Console 之外与 Amazon 交互，则需要编程式访问权限。Amazon API 和 Amazon Command Line Interface 需要访问密钥。可能的话，创建临时凭证，该凭证由一个访问密钥 ID、一个秘密访问密钥和一个指示凭证何时到期的安全令牌组成。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
IAM	使用短期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中将临时凭证用于 Amazon 资源中的说明进行操作。

哪个用户需要编程式访问权限？	目的	方式
IAM	（不推荐使用） 使用长期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中管理 IAM 用户的访问密钥中的说明进行操作。

下一个步骤

步骤 2：设置 Amazon Command Line Interface (Amazon CLI) (p. 89)

步骤 2：设置 Amazon Command Line Interface (Amazon CLI)

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

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《Amazon Command Line Interface用户指南》Amazon Command Line Interface中的安装。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。有关命名配置文件的更多信息，请参阅Amazon Command Line Interface用户指南中的命名配置文件。

   ```
   [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Amazon区域的列表，请参阅 Amazon Web Services 科技一般引用中的区域和终端节点。

   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
```

在您设置Amazon帐户后Amazon CLI，您可以尝试下一个练习，在其中配置示例应用程序并测试 end-to-end 设置。

下一个步骤

步骤 3：为 Apache Flink 应用程序创建、运行 Kinesis Data Analytics for (p. 90)

步骤 3：为 Apache Flink 应用程序创建、运行 Kinesis Data Analytics for

在本练习中，您将创建一个以数据流作为源和汇点的 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 应用程序之前，请创建两个 Kinesis 数据流（`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Amazon CLI命令创建这些直播。有关控制台的说明，请参阅 Amazon Kinesis 数据流开发者指南中的创建和更新数据流。

创建数据流 (Amazon CLI)

1. 要创建第一个直播 (`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', region_name='us-west-2'))
```

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 应用程序代码可从中获得 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：

- 项目对象模型 (pom.xml) 文件包含有关应用程序配置和依赖关系的信息，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 源从源码流中读取数据。以下片段创建了 Kinesis 源代码：

```
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. 25)。

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. 87)。

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=1.15.3
     ```

   - 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。

   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 Simple Storage Simple Simple Simple Simple Simple

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. **`ka-app-code-<username>`**在存储段名称字段中输入。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. 选择创建桶。
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。选择下一步。
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，您的应用程序可以在其中访问。

创建并运行 Kinesis Data Analytics 应用程序

您可以使用控制台或，创建和运行 Kinesis Data AnalyticsAmazon CLI

> Note
>
> 当您使用控制台创建应用程序时，会为您创建Amazon Identity and Access Management (IAM) 和 A CloudWatch mazon Logs 资源。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，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。
   - 将版本下拉列表保留为 Apache Flink 版本 1.15.2（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 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",
                "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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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：

组 ID	键	值
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
8. 选择更新。

> **Note**
>
> 当您选择启用亚马逊 CloudWatch 日志记录时，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 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

停止应用程序

在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，也可以从 Amazon S3 存储桶重新加载应用程序 JAR。

在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)

在本节中，您将使用创建和运行 Kinesis Data Analytics 应用程序。Amazon CLI适用于 Apache Flink 的 Kinesis Data Analytics 使用该`kinesisanalyticsv2`Amazon CLI命令创建 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（您在下一节中创建）。因此，当 Kinesis Data Analytics 担任该角色时，该服务具有从源流读取和写入接收流的必要权限。

使用以下代码创建 KAReadSourceStreamWriteSinkStream 权限策略。*username*替换为用于创建 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"
        }
    ]
}
```

有关创建权限策略的 step-by-step 说明，请参阅 IAM 用户指南中的教程：创建并附加您的第一个客户托管策略。

Note

要访问其他亚马逊服务，您可以使用Amazon SDK for Java。Kinesis Data Analytics 会自动将软件开发工具包所需的证书设置为与您的应用程序关联的服务执行 IAM 角色的证书。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 应用程序可以假定该角色来读取源流并写入接收流。

未经许可，Kinesis Data Analytics 无法访问您的直播。您可以通过 IAM 角色授予这些权限。每个 IAM 角色都附加了两个策略。信任策略授予 Kinesis Data Analytics（Kinesis Data Analytics）代入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 “选择可信身份类型” 下，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 Attach permissions policies 页面上，选择 Next: Review。在创建角色后，您可以附加权限策略。
5. 在创建角色页面上**`KA-stream-rw-role`**，输入角色名称。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 Analytics 扮演这个角色，既可以从 Kinesis 数据流（源）读取数据，也可以将输出写入另一个 Kinesis 数据流。因此，您附加在上一步（the section called “创建权限策略” (p. 96)）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。

有关创建角色的 step-by-step 说明，请参阅 IAM 用户指南中的创建 IAM 角色（控制台）。

创建 Kinesis Data Analytics

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_15",
    "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 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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 日志选项

您可以使用Amazon CLI向您的应用程序添加 Amazon CloudWatch 日志流。有关在应用程序中使用 CloudWatch Logs 的信息，请参阅the section called “设置日志记录” (p. 269)。

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。在此示例中，您更改了源直播流和目标直播的区域。

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

要加载具有相同文件名的新版本的应用程序代码，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。

要使用Amazon CLI，请从 Amazon S3 存储桶中删除您之前的代码包，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. 90)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

步骤 4：清除 “Amazon资源” (p. 100)

步骤 4：清除 “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 (p. 101)
- 删除 Kinesis Data Streams (p. 101)
- 删除您的Amazon S3 对象和存储桶 (p. 101)
- 删除您的 IAM 资源 (p. 101)
- 删除您的 CloudWatch 资源 (p. 101)

- 下一个步骤 (p. 101)

删除 Kinesis Data Analytics

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，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream。
3. 在ExampleInputStream页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择 “操作”，选择 “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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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

步骤 5：后续步骤 (p. 102)

步骤 5：后续步骤

现在，您已经创建并运行了基本的 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。

- Amazon Kinesis 的Amazon流媒体数据解决方案：Amazon Kinesis 的流媒体数据解决方案可自动配置必要的Amazon服务，以便轻松捕获、存储、处理和传输流数据。Amazon该解决方案为解决流数据用例提供了多个选项。Kinesis Data Analytics 选项提供了一个 end-to-end流式传输 ETL 示例，演示了对模拟的纽约出租车数据进行分析操作的真实应用程序。该解决方案设置了所有必需的Amazon资源，例如 IAM 角色和策略、 CloudWatch 仪表板和 CloudWatch 警报。
- AmazonAmazon MSK 的Amazon流媒体数据解决方案：Amazon MSK 的流媒体数据解决方案提供了数据流经生产者、流媒体存储、消费者和目的地的Amazon CloudFormation模板。
- 带有 Apache Flink 和 Apache Kafka 的 Clickst ream Lab：一个端到端的点击流实验室，使用适用于AManaged Streaming for Apache Kafka 的 Amazon Managed Streaming 进行流存储，使用适用于 Apache Flink 的Amazon Kinesis Data Analytics 应用程序进行流处理。
- 流媒体分析研讨会：在本研讨会中，您将构建一个 end-to-end 流媒体架构，以近乎实时地提取、分析和可视化流媒体数据。你着手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队的遥测数据，以优化其车队运营。
- 适用于 Apach Flink (p. 146)：本开发者指南的这一部分提供了在 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，可帮助您创建 Kinesis Data Analytics 应用程序和测试结果。
- 学习 Flink：动手训练：Apache Flink 官方入门培训，让你开始编写可扩展的流媒体 ETL、分析和事件驱动应用程序。

  Note

  请注意，Kinesis Data Analytics 不支持本次培训中使用的 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 (Amazon

本节向你介绍适用于 Apache Flink 的 Kinesis Data Analytics 和 Table 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

适用于 Flink 应用程序的 Kinesis Data Analytics 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 运行时处理输入并生成输出。

Kinesis Data Analytics 应用程序包含以下组件：

- 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。
- 表源：应用程序使用源来消耗数据。源连接器从 Kinesis 数据流、亚马逊 MSK 主题或类似主题读取数据。有关更多信息，请参阅表 API 源代码 (p. 19)：
- 函数：应用程序使用一个或多个函数处理数据。函数可以转换、丰富或聚合数据。
- S@@ ink：应用程序使用接收器向外部源生成数据。接收器连接器将数据写入 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. 20)：

创建、编译和打包应用程序代码后，您将文件包上载到 Amazon S3 桶。然后，您将创建一个 Kinesis Data Analytics 应用程序。您传入代码包位置，将 Amazon MSK 主题作为流式传输数据源，通常是接收应用程序处理数据的流式传输或文件位置。

先决条件

在开始本教程之前，请完成前两个步骤Amazon Kinesis for Apache Flink (DataStream API) (p. 87)：

- 步骤 1：设置 Amazon 账户并创建管理员用户 (p. 88)
- 步骤 2：设置 Amazon Command Line Interface (Amazon CLI) (p. 89)

要开始使用，请参阅创建应用程序 (p. 104)。

创建并运行 Apache Flink 应用程序的 Kinesis Data Analytics

在本练习中，您将创建 Kinesis Data Analytics 应用程序，将 Amazon MSK 主题作为来源，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

在为本练习创建适用于 Apache Flink 的 Amazon Kinesis 数据分析之前，您需要创建以下依赖资源：

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

创建 VPC 和亚马逊 MSK 集群

要创建从您的 Kinesis Data Analytics 应用程序访问的 VPC 和 Amazon MSK 集群，请按照亚马逊 MSK 入门教程进行操作。

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，*ClusterArn*替换为 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"
}
```

- 按照教程中的步骤进行操作时，请务必在代码、命令和控制台条目中使用所选Amazon区域。

创建 Amazon S3 存储桶

您可以使用控制台来创建 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？ 在 Amazon Simple Storage Serv ice 通过附加您的登录名，为 Amazon S3 存储桶指定一个全球唯一的名称，例如**ka-app-code-*<username>***。

其他资源

在创建应用程序时，Kinesis Data Analytics 将创建以下Amazon CloudWatch 资源（如果尚不存在）：

- 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，您将使用 Python 脚本将示例记录写入 Amazon MSK 主题以供应用程序处理。

1. Connect 您在使用 Amazon MSK 入门教程的步骤 4：创建客户端计算机中创建的客户端实例。
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。用你之前记录的引导代理列表替换该BROKERS值。

```
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', region_name='us-west-2'))
```

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 应用程序代码可从中获得 GitHub。

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：

- 项目对象模型 (pom.xml) 文件包含有关应用程序配置和依赖关系的信息，包括 Kinesis Data Analytics 库。
- 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 aFlinkKafkaConsumer 来读取 Amazon 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StreamExecutionEnvironment和TableEnvironment对象创建源和接收器连接器，以访问外部资源。
- 该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，因此您无需重新编译代码即可指定应用程序参数（例如 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. 25)。

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. 87)。

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=1.15.3
```

- 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。

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

在本部分中，您将创建一个 Service 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. **ka-app-code-*<username>***在存储段名称字段中输入。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. 选择创建桶。
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。选择下一步。
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该存储桶。

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，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。
   - 将该版本保留为 Apache Flink 版本 1.15.2（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 策略以添加权限以添加权限以添加访问 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": "S3",
        "Effect": "Allow",
        "Action": [
            "s3:Abort*",
            "s3:DeleteObject*",
            "s3:GetObject*",
            "s3:GetBucket*",
            "s3:List*",
            "s3:ListBucket",
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::ka-app-code-<username>",
            "arn:aws:s3:::ka-app-code-<username>/*"
        ]
        },
        {
            "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"
            ]
        }
    ]
}
```

配置应用程序

请使用以下过程配置应用程序。

配置应用程序

1. 在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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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 “属性” 下，选择 “创建群组”。
5. 输入以下信息:

组 ID	键	值
FlinkApplicationProperties	**kafka-topic**	**AWSKafkaTutorialTopic**
FlinkApplicationProperties	**brokers**	***Your Amazon MSK cluster's Bootstrap Brokers list***
FlinkApplicationProperties	**s3Path**	**ka-app-code-*<username>***
FlinkApplicationProperties	**security.protocol**	**SSL**
FlinkApplicationProperties	**ssl.truststore.location**	**/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts**
FlinkApplicationProperties	**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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
8. 在V irtual Private Cloud (VPC) 部分中，选择基于 Amazon MSK 集群的 VPC 配置。选择AWSKafkaTutorialCluster。
9. 选择更新。

Note

当您选择启用亚马逊 CloudWatch 日志记录时，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

请使用以下过程运行应用程序。

运行应用程序

1. 在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)。
3. 在您的 Amazon EC2 客户端上，运行您之前创建的 Python 脚本，将记录写入 Amazon MSK 集群以供应用程序处理：

```
$ python3 stock.py
```

停止应用程序

要停止应用程序，请在MyApplication页面上选择停止。确认该操作。

下一个步骤

清理 Amazon 资源 (p. 110)

清理 Amazon 资源

本节包括清理入门（Table API）教程中创建的Amazon资源的过程。

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，选择删除，然后确认删除。

删除您的亚马逊 MSK 集群

要删除您的 Amazon MSK 集群，请按照适用于 Apache Kafka 的 Amazon MSK 流媒体管理开发者指南中的步骤 8：删除亚马逊 MSK 集群。

删除您的 VPC

要删除您的 Amazon VPC，请执行以下操作：

- 打开 Amazon VPC 控制台。
- 选择您的 VPC。
- 对于 Actions (操作)，请选择 Delete VPC (删除 VPC)。

删除您的 Amazon S3 对象和存储桶

请使用以下过程删除您的 S3 对象和存储桶。

删除您的 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

请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>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 k inesis-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

请使用以下过程删除 CloudWatch 资源。

删除您的 CloudWatch 资源

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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. 112)

后续步骤

现在，您已经创建并运行了使用 Table API 的 Kinesis Data Analytics 应用程序，请参见步骤 5：后续步骤 (p. 102)中的Amazon Kinesis for Apache Flink (DataStream API) (p. 87)。

Amazon Kinesis Data Analytics for Apache

本节向你介绍使用 Python 和 Table API 的 Apache Flink 版 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

在开始之前，我们建议您观看以下视频：

Pyflink 入门——适用于 Apache 的 Python 解释器 | Amazon Web Services

适用于 Flink 应用程序的 Kinesis Data Analytics 组件

为了处理数据，您的 Kinesis Data Analytics 应用程序使用 Python 应用程序，该应用程序使用 Apache Flink 运行时处理输入并生成输出。

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。
- 表源：应用程序使用源来消耗数据。源连接器从 Kinesis 数据流、亚马逊 MSK 主题或类似主题读取数据。有关更多信息，请参阅表 API 源代码 (p. 19)：
- 函数：应用程序使用一个或多个函数处理数据。函数可以转换、丰富或聚合数据。
- S@@ ink：应用程序使用接收器向外部源生成数据。接收器连接器将数据写入 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. 20)：

在创建和打包应用程序代码时，您将代码包上载到 Amazon S3 存储桶。然后，您将创建一个 Kinesis Data Analytics 应用程序。您传入代码包位置、流数据源，通常是接收应用程序处理数据的流媒体或文件位置。

先决条件

在开始本教程之前，请完成前两个步骤Amazon Kinesis for Apache Flink (DataStream API) (p. 87)：

- 步骤 1：设置 Amazon 账户并创建管理员用户 (p. 88)
- 步骤 2：设置 Amazon Command Line Interface (Amazon CLI) (p. 89)

要了解其用法，请参阅 创建应用程序 (p. 114)。

为 Python 应用程序创建并运行 Kinesis Data Analytics

在本练习中，您将为 Python 应用程序创建 Kinesis Data Analytics 应用程序，将 Kinesis 流作为源和接收器。

本节包含以下步骤。

- 创建相关资源 (p. 114)
- 将示例记录写入输入流 (p. 115)
- 创建并检查 Apache Flink 直播 Python 代码 (p. 116)
- 上传 Apache Flink 直播 Python 代码 (p. 117)
- 创建并运行 Kinesis Data Analytics 应用程序 (p. 118)
- 下一个步骤 (p. 121)

创建相关资源

在为本练习创建适用于 Apache Flink 的 Amazon Kinesis 数据分析之前，您需要创建以下依赖资源：

- 两个 Kinesis 数据流用于输入和输出。
- 用于存储应用程序代码和输出的 Amazon S3 存储桶 (ka-app-code-*<username>*)

创建两个 Kinesis 直播

在为本练习创建 Kinesis Data Analytics 应用程序之前，请创建两个 Kinesis 数据流（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Amazon CLI命令创建这些直播。有关控制台的说明，请参阅 Amazon Kinesis 数据流开发者指南中的创建和更新数据流。

创建数据流 (Amazon CLI)

1. 要创建第一个直播 (ExampleInputStream)，请使用以下 Amazon Kinesiscreate-streamAmazon 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 Amazon S3 存储桶

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

- 如何创建 S3 存储桶？ 在 Amazon Simple Storage Serv ice 通过附加您的登录名，为 Amazon S3 存储桶指定一个全球唯一的名称，例如**ka-app-code-*<username>***。

其他资源

在创建应用程序时，Kinesis Data Analytics 将创建以下Amazon CloudWatch 资源。

- 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. 您必须将您的配置Amazon CLI为使用您的账户凭证和默认区域。要配置您的Amazon CLI，请输入以下内容：

```
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', region_name='us-west-2'))
```

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。

创建并检查 Apache Flink 直播 Python 代码

此示例的 Python 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. 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：

- 该应用程序使用 Kinesis 表源从源数据流中读取数据。以下代码段调用该create_table函数来创建 Kinesis 表源：

```
table_env.execute_sql(
        create_table(output_table_name, output_stream, output_region)
```

该create_table函数使用 SQL 命令创建由流式传输源支持的表：

```
def create_table(table_name, stream_name, region, stream_initpos = None):
    init_pos = "\n'scan.stream.initpos' = '{0}',".format(stream_initpos) if
 stream_initpos is not None else ''

    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}',{3}
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(table_name, stream_name, region, init_pos)
 }
```

- 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)
)

# 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))
```

- 该应用程序使用来自 flink - sql-connector-kinesis _2.12/1.15.2 文件的 Flink 连接器。
- 使用第三方 python 软件包（例如 boto3）时，需要将其添加到`getting-started.py`所在的GettingStarted文件夹中。无需在 Apache Flink 或 Kinesis Data Analytics 中添加任何其他配置。可以在如何在 PyfLink 中使用 boto3 中找到一个示例。

上传 Apache Flink 直播 Python 代码

在本部分，您将创建一个 Simple Storage Service（Amazon S3）存储桶并上传应用程序

要使用控制台上传应用程序代码，请执行以下操作：

1. 使用你首选的压缩应用程序压缩`getting-started.py`和 https://mvnrepository.com/artifact/org.apache.flink/flink- sql-connector-kinesis _2.12/1.15.2 文件。命名档案`myapp.zip`。如果您在存档中包含外部文件夹，则必须将其包含在配置文件中的代码路径中：`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. **`ka-app-code-<username>`**在存储段名称字段中输入。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. 选择创建桶。
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)。

要使用以下命令上传应用程序代码，请执行以下Amazon CLI操作：

> **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。这可能导致应用程序代码无效。

1. 使用你首选的压缩应用程序压缩`streaming-file-sink.py`和 https://mvnrepository.com/artifact/org.apache.flink/flink- sql-connector-kinesis _2.12/1.15.2 文件。

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。这可能导致应用程序代码无效。

2. 使用您首选的压缩应用程序压缩`getting-started.py`和 https://mvnrepository.com/artifact/org.apache.flink/flink-sql-connector-kinesis /1.15.2 文件。命名档案`myapp.zip`。如果您在存档中包含外部文件夹，则必须将其包含在配置文件中的代码路径中：`GettingStarted/getting-started.py`。
3. 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，您的应用程序可以在该存储桶中进行访问。

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，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。
   - 将该版本保留为 Apache Flink 版本 1.15.2 (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`

配置应用程序

可以按照以下过程来配置应用程序。

配置应用程序

1. 在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)：

- 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
- 在 Amazon S3 对象的路径中，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. 在 “属性” 下，选择 “添加群组”。
5. 输入以下信息:

组 ID	键	值
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

选择保存。

6. 在 “属性” 下，再次选择 “添加群组”。
7. 输入以下信息:

组 ID	键	值
producer.config.0	**output.stream.name**	**ExampleOutputStream**
producer.config.0	**aws.region**	**us-west-2**
producer.config.0	**shard.count**	**1**

8. 在 “属性” 下，再次选择 “添加群组”。对于群组 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，请参阅指定您的代码文件 (p. 23)：
9. 输入以下信息:

组 ID	键	值
kinesis.analytics.flink.run.options	**python**	**getting-started.py**
kinesis.analytics.flink.run.options	**jarfile**	**flink-sql-connector-kinesis-1.15.2.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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
12. 选择更新。

Note

当您选择启用亚马逊 CloudWatch 日志记录时，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`

编辑 IAM 策略

编辑 IAM 策略以添加访问 Amazon S3 存储桶的权限。

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

可以通过运行应用程序、打开 Apache Flink 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

停止应用程序

要停止应用程序，请在MyApplication页面上选择停止。确认该操作。

下一个步骤

清理 Amazon 资源 (p. 121)

清理 Amazon 资源

本节包括清理入门 (Python) 教程中创建的Amazon资源的过程。

本主题包含以下部分。

删除 Kinesis Data Analytics

可以按照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，选择删除，然后确认删除。

删除 Kinesis Data Streams

1. 通过 https://console.aws.amazon.com/kinesisanalytics 打开 Kinesis Data Analytics 控制台。

2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream。
3. 在ExampleInputStream页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis 直播页面中 ExampleOutputStream，选择，选择操作，选择删除，然后确认删除。

删除您的 Amazon S3 对象和存储桶

可以按照以下过程删除Simple Storage（ Simple ）。

删除您的 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

可以按照以下过程删除您的 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>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 k inesis-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

可以按照以下过程删除 CloudWatch 资源。

删除您的 CloudWatch 资源

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

入门 (Scala)

> Note
>
> 从 1.15 版本开始，Flink 是免费的 Scala。应用程序现在可以使用任何 Scala 版本的 Java API。Flink 仍然在一些关键组件中使用 Scala，但不会将 Scala 暴露给用户代码类加载器。因此，用户需要将 Scala 依赖项添加到他们的 jar 存档中。
> 有关 Flink 1.15 中 Scala 变更的更多信息，请参阅 One Feifteen 中的 Scala Free。

在本练习中，您将创建 Scala 的 Kinesis Data Analytics 应用程序应用程序，将 Kinesis 流作为源和接收器。

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，您需要创建以下依赖资源：

- 两个 Kinesis 数据流用于输入和输出。
- 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 Streams 和 Amazon S3 存储桶。有关创建这些资源的说明，请参阅以下主题：

- 在 Amazon Kinesis Data St reams Data Stream s？ 将数据流命名为 **`ExampleInputStream`** 和 **`ExampleOutputStream`**。

  创建数据流 (Amazon CLI)

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

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

  - 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
```

- 如何创建 S3 存储桶？ 在 Amazon Simple Storage 用户指南中。通过附加您的登录名，为 Amazon S3 存储桶指定一个全球唯一的名称，例如**ka-app-code-*<username>***。

其他资源

在创建应用程序时，Kinesis Data Analytics 将创建以下Amazon CloudWatch 资源（如果尚不存在）：

- 一个名为的日志组`/aws/kinesis-analytics-java/MyApplication`
- 名为的日志流`kinesis-analytics-log-stream`

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. 您必须将您的配置Amazon CLI为使用您的账户凭证和默认区域。要配置您的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', region_name='us-west-2'))
```

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

此示例的 Python 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. 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/scala/GettingStarted目录。

请注意有关应用程序代码的以下信息：

- build.sbt文件包含有关应用程序配置和依赖关系的信息，包括 Kinesis Data Analytics 库。
- 该BasicStreamingJob.scala文件包含定义应用程序功能的主要方法。
- 该应用程序使用 Kinesis 源从源码流中读取数据。以下片段创建了 Kinesis 源代码：

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

该应用程序还使用 Kinesis 接收器来写入结果流。以下片段创建了 Kinesis 水槽：

```
private def createSink: KinesisStreamsSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val outputProperties = applicationProperties.get("ProducerConfigProperties")

  KinesisStreamsSink.builder[String]
    .setKinesisClientProperties(outputProperties)
    .setSerializationSchema(new SimpleStringSchema)
    .setStreamName(outputProperties.getProperty(streamNameKey, defaultOutputStreamName))
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))
    .build
}
```

- 应用程序创建源和接收连接器以使用 StreamExecutionEnvironment 对象访问外部资源。
- 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。读取运行时应用程序的属性以配置连接器。有关运行时属性的更多信息，请参见运行时属性。

编译并上传应用程序代码

在本节中，您将编译应用程序代码并将其上传到您在该创建相关资源 (p. 123)部分创建的 Amazon S3 存储桶。

编译应用程序代码

在本节中，您将使用 SBT 构建工具为应用程序构建 Scala 代码。要安装 SBT，请参阅使用 cs 设置安装 sbt。您还需要安装 Java 开发工具包 (JDK)。请参阅完成练习的先决条件。

1. 要使用您的应用程序代码，您将其编译和打包成 JAR 文件。你可以用 SBT 编译和打包你的代码：

```
sbt assembly
```

2. 如果应用程序成功编译，则创建以下文件：

```
target/scala-3.2.0/getting-started-scala-1.0.jar
```

上传 Apache Flink 直播 Scala 代码

在此部分，您将创建一个 Simple Storage S3 存储桶并上传应用程序代码。

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择创建存储桶
3. ka-app-code-<username>在存储段名称字段中输入。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。选择下一步。
4. 在配置选项中，保持设置不变，然后选择下一步。
5. 在设置权限中，保持设置不变，然后选择下一步。
6. 选择创建桶。
7. 选择该ka-app-code-<username>存储桶，然后选择 Upload（上载）。
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 getting-started-scala-1.0.jar 文件。
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应用程序。

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 scala test app**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。
   - 将该版本保留为 Apache Flink 版本 1.15.2（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*

配置应用程序

可以按照以下步骤配置应用程序。

配置应用程序

1. 在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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，输入**getting-started-scala-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 “属性” 下，选择 “添加群组”。
5. 输入以下信息:

组 ID	键	值
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

   选择保存。
6. 在 “属性” 下，再次选择 “添加群组”。
7. 输入以下信息:

组 ID	键	值
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
10. 选择更新。

Note

当您选择启用亚马逊 CloudWatch 日志记录时，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`

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/getting-started-scala-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"
        }
    ]
}
```

运行应用程序

可以通过运行应用程序、打开 Apache Flink 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

停止应用程序

要停止应用程序，请在MyApplication页面上选择停止。确认该操作。

创建并运行应用程序 (CLI)

在本节中，您将使用创建和运行 Kinesis Data Analytics 应用程序。Amazon Command Line Interface使用 kinesisanalyticsv2Amazon CLI 命令创建 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，您创建一个包含两个语句的权限策略：一个语句授予对源流进行读取操作的权限，另一个语句授予对接收流进行写入操作的权限。然后，您将策略附加到 IAM 角色（您在下一节中创建）。因此，当 Kinesis Data Analytics 担任该角色时，该服务具有从源流读取和写入接收流的必要权限。

使用以下代码创建 KAReadSourceStreamWriteSinkStream 权限策略。**username**替换为用于创建 Amazon S3 存储桶以存储应用程序代码的用户名。将亚马逊资源名称 (ARN) 中的账户 ID 替换**(012345678901)**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/getting-started-scala-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"
        }
    ]
}
```

有关创建权限策略的 step-by-step 说明，请参阅 IAM 用户指南中的教程：创建并附加您的第一个客户托管策略。

创建 IAM policy

在本节中，您将创建一个 IAM 角色，Kinesis Data Analytics 应用程序可以假定该角色来读取源流并写入接收流。

未经许可，Kinesis Data Analytics 无法访问您的直播。您可以通过 IAM 角色授予这些权限。每个 IAM 角色都附加了两个策略。信任策略授予 Kinesis Data Analytics 代入角色的权限，权限策略决定了Kinesis Data Analytics 代入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，选择角色，然后选择创建角色。
3. 在 “选择可信身份类型” 下，选择 “Amazon服务”
4. 在 Choose the service that will use this role (选择将使用此角色的服务) 下，选择 Kinesis。
5. 在选择您的使用案例下，选择 Kinesis Analytics。
6. 选择Next: Permissions（下一步: 权限）。
7. 在 Attach permissions policies 页面上，选择 Next: Review。在创建角色后，您可以附加权限策略。
8. 在创建角色页面上**KA-stream-rw-role**，输入角色名称。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，更新角色的信任和权限策略
9. 将权限策略附加到角色。

   Note

   在本练习中，Kinesis Data Analytics 扮演这个角色，既可以从 Kinesis 数据流（源）读取数据，也可以将输出写入另一个 Kinesis 数据流。因此，您可以附加在上一步中创建一个 “创建权限策略” 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。

有关创建角色的 step-by-step 说明，请参阅 IAM 用户指南中的创建 IAM 角色（控制台）。

创建应用程序

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（用户名）替换为您在上一节中选择的后缀。将服务执行角色中的示例账户 ID (012345678901) 替换为您的账户 ID。

```
{
    "ApplicationName": "getting_started",
    "ApplicationDescription": "Scala getting started application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/KA-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "getting-started-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
```

```
                "PropertyGroupId": "ConsumerConfigProperties",
                "PropertyMap" : {
                     "aws.region" : "us-west-2",
                     "stream.name" : "ExampleInputStream",
                     "flink.stream.initpos" : "LATEST"
                }
            },
            {
                "PropertyGroupId": "ProducerConfigProperties",
                "PropertyMap" : {
                     "aws.region" : "us-west-2",
                     "stream.name" : "ExampleOutputStream"
                }
            }
        ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-group:MyApplication:log-
stream:kinesis-analytics-log-stream"
      }
   ]
}
```

CreateApplication使用以下请求执行以创建应用程序：

```
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": "getting_started",
    "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 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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": "s3_sink"
}
```

2. 使用上述请求执行StopApplication操作以停止应用程序：

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

应用程序现已停止。

添加 CloudWatch 日志选项

您可以使用Amazon CLI向您的应用程序添加 Amazon CloudWatch 日志流。有关在应用程序中使用 CloudWatch 日志的信息，请参阅设置应用程序日志。

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。在此示例中，您将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": "getting_started",
       "CurrentApplicationVersionId": 1,
       "ApplicationConfigurationUpdate": {
        "EnvironmentPropertyUpdates": {
           "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
           ]
        }
    }
```

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

当您需要使用新版本的代码包更新应用程序代码时，可以使用 UpdateApplicationCLI 操作。

Note

要加载具有相同文件名的新版本的应用程序代码，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。

要使用Amazon CLI，请从 Amazon S3 存储桶中删除您之前的代码包，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。<username>使用您在部分中选择的后缀更新存储创建相关资源 (p. 123)段名称后缀 ()。

```
{{
    "ApplicationName": "getting_started",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-<username>",
                    "FileKeyUpdate": "getting-started-scala-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

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：

- 删除您的 Kinesis Data Analytics 应用程序 (p. 134)
- 删除您的 Kinesis Data Streams (p. 135)
- 删除您的Amazon S3 对象和存储桶 (p. 135)
- 删除您的 IAM 资源 (p. 135)
- 删除您的 CloudWatch 资源 (p. 135)

删除您的 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。

删除您的 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页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择 “操作”，选择 “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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

使用 Apache Beam 创建 Kinesis Data Analytics 应用程序

您可以将 Apache Beam 框架与 Kinesis Data Analytics 应用程序一起使用来处理流数据。使用 Apache Beam 的 Kinesis Data Analytics 应用程序使用 Apache Flink runner 来执行 Beam 流水线。

有关如何在 Kinesis Data Analytics 应用程序中使用的教— 示— 示— 示— 示— 示— 示— 示— 示— 示 CloudFormation 与 Kinesis Data Analytics (p. 136) — 示

本主题包含下列部分：

- 将 Apache Beams 与 Kinesis Data (p. 136)
- 光束能力 (p. 136)
- 使用 Apache Beam 创建应用程序 (p. 136)

将 Apache Beams 与 Kinesis Data

关于将 Apache Flink 运行器与 Kinesis Data Analytics 一起使用，请注意以下几点：

- 无法在 Kinesis Data Analytics 控制台中查看 Apache Beam 指标。
- 只有使用 Apache Flink 版本 1.8 及更高版本的 Kinesis Data Analytics 应用程序支持 Apache Beam。使用 Apache Flink 版本 1.6 的 Kinesis Data Analytics 应用程序不支持 Apache Beam。

光束能力

Kinesis Data Analytics 支持与 Apache Flink 运行器相同的 Apache Beam 功能。有关 Apache Flink 运行器支持哪些功能的信息，请参阅 Bea m 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，以验证我们是否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，您将创建一个使用 Apache Beam 转换数据的 Kinesis Data Analytics 应用程序。Apache Beam 是一种用于处理流数据的编程模型。有关在 Kinesis Data Analytics 中使用 Apache Beam 的信息，请参阅使用Apache Beam (p. 136)。

> Note
>
> 要为本练习设置所需的先决条件，请先完成入门指南 (DataStream API) (p. 87)练习。

本主题包含下列部分：

- 创建相关资源 (p. 137)
- 将示例记录写入输入流 (p. 137)
- 下载并检查应用程序代码 (p. 137)
- 编译应用程序代码 (p. 138)

创建相关资源

在为本练习创建 Kinesis Data Analytics 应用程序之前，您需要创建以下依赖资源：

- 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 () 存储桶 (`ka-app-code-<username>`)

您可以使用控制台创建 Kinesis Streams 和 Amazon S3 存储桶。有关创建这些资源的说明，请参阅以下主题：

- Amazon Kinesis Data Streams？ 将数据流命名为 **`ExampleInputStream`** 和 **`ExampleOutputStream`**。
- 如何创建 S3 存储桶？ Amazon Simple Storage 用户指南。通过附加您的登录名，为 Amazon S3 存储桶指定一个全球唯一的名称，例如**`ka-app-code-<username>`**。

将示例记录写入输入流

在本节中，您将使用 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

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. 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 Beam 通过调用名ParDo为的自定义转换函数来处理传入的记录PingPongFn。

  调用该PingPongFn函数的代码如下所示：

```
.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。如果您不在您的pom.xml中包含这些组件和版本，则您的应用程序会从环境依赖项中加载错误的版本，并且由于版本不匹配，您的应用程序会在运行时崩溃。

```
<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>
```

- PingPongFn转换函数将输入数据传递到输出流中，除非输入数据为 ping，在这种情况下，它会向输出流发出字符串 pong\ n。

  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，请参阅入门指南 (DataStream API) (p. 87)教程中的先决条件 (p. 87)。
2. 使用以下命令编译应用程序：

```
mvn package -Dflink.version=1.15.3 -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

在本部分中，您将应用程序代码上传到您在该创建相关资源 (p. 137)部分创建的 Amazon S3 存储桶。

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 `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存储在 Amazon S3 存储桶中，应用程序可以在此存储桶中进行访问。

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，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 1.15.2。
   - 将版本下拉列表保留为 Apache Flink 版本 1.15.2 (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 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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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. 输入以下信息:

组 ID	键	值
BeamApplicationProperties	**InputStreamName**	**ExampleInputStream**
BeamApplicationProperties	**OutputStreamName**	**ExampleOutputStream**
BeamApplicationProperties	**AwsRegion**	**us-west-2**

5. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
6. 要进行CloudWatch 记录，请选中 “启用” 复选框。
7. 选择更新。

Note

当您选择启用 CloudWatch 日志记录时，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`

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 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

您可以在 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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

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页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择 “操作”，选择 “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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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 解决方案的示例。

- Beam on Kinesis Data Analytics 直播研讨会：在本次研讨会中，我们将探讨一个端到端的示例，该示例将批处理和串流方面结合在一个统一的 Apache Beam 流水线中。

培训研讨会、实验室和解决方案实施

以下 end-to-end 示例演示了高级 Kinesis Data Analytics 解决方案。

主题

在部署到 Apache Flink Data Analytics for Apache Flink Analytics

本次研讨会将向你展示基础知识，并开始在本地开发 Apache Flink 应用程序，长期目标是部署到 Kinesis Data Analytics for Apache Flink。

解决方案可以在这里找到：使用 Apache Flink 进行本地开发的入门指南

使用 Kinesis Data Analytics

本研讨会介绍了使用 Kinesis Data Analytics 工作室进行事件检测并将其部署为 Kinesis Data Analytics 应用程序

解决方案可以在这里找到：使用适用于 Apache Flink 的 Kinesis Data Analytics 进行事件检测

AmazonAmazon Kinesis 的流媒体数据解决方案

Amazon Kinesis 的Amazon流媒体数据解决方案可自动配置必要的Amazon服务，以便轻松捕获、存储、处理和传输流数据。该解决方案为解决流数据用例提供了多个选项。Kinesis Data Analytics 选项提供了一个 end-to-end 流式传输 ETL 示例，演示了对模拟的纽约出租车数据进行分析操作的真实应用程序。

每个解决方案包含以下组件：

- 用于部署完整示例的Amazon CloudFormation软件包。
- 用于显示应用程序指标的 CloudWatch 仪表板。
- CloudWatch 有关最相关的应用程序指标的警报。
- 所有必要的 IAM 角色和策略。

解决方案可以在这里找到：Amazon Kinesis 的流媒体数据解决方案

带有 Apache Flink 和 Apache Kafka 的 Click

点击流用例的端到端实验室，使用适用于 AManaged Streaming for Apache Kafka 的 Amazon Managed Streaming 进行流存储，使用适用于 Apache Flink 的 Amazon Kinesis 数据分析应用程序进行流处理。

解决方案可以在这里找到：Clickstream Lab

使用Auto Auto Scalal

该示例可帮助用户使用应用程序自动扩展Application Auto Scaling 于 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。

解决方案可以在这里找到：

- Kinesis Data Analytics Flink 应用程序自动缩放
- 定时扩展

亚马逊 CloudWatch 控制面板

用于监控Amazon Kinesis Data Analytics 应用程序的示例 CloudWatch 控制面板。示例仪表板还包括一个演示应用程序，以帮助演示仪表板的功能。

解决方案可以在这里找到：Kinesis Data Analytics 指标仪表板

Amazon亚马逊 MSK 的流媒体数据解决方案

Amazon MSK 的Amazon流媒体数据解决方案提供了Amazon CloudFormation模板，其中数据流经生产者、流媒体存储、消费者和目的地。

解决方案可以在这里找到：亚马逊 MSK 的Amazon流媒体数据解决方案

更多 Kinesis Data Analytics 解决方案请访问 GitHub

以下 end-to-end 示例演示了高级 Kinesis Data Analytics 解决方案，可在以下平台上使用 GitHub：

- Amazon Kinesis Data Analytics Flink — 基准测试工具
- 快照管理器 — Amazon Kinesis Data Analytics for A
- 使用 Apache Flink 和Amazon Kinesis Data Analytics 直播 ETL
- 对客户反馈进行实时情绪分析

实用程序

以下实用程序可以更轻松地使用 Flink 版 Kinesis Data Analytics 服务：

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/快照以实现更无缝的故障恢复。Snapsot Manalyter 自动执行此任务并提供以下优势：

- 拍摄正在运行的 Kinesis Data Analytics for Apache Flink
- 获取应用程序快照的数量
- 检查计数是否超过所需的快照数量
- 删除比所需数量更早的旧快照

有关示例，请参见 Flink 的快照管理器

基准测试

Kinesis Data Analytics Flink 基准测试实用程序可帮助对 Apache Flink 应用程序的 Kinesis Data Analytics 进行容量规划、集成测试和基准测试。

有关示例，请参阅基准测试

适用于 Apach Flink

本节提供了在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，可帮助您创建 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)
- 入门指南 (DataStream API) (p. 87)

> 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 (p. 146)
- Python 示例 (p. 199)
- Scala 示例 (p. 219)

DataStream API 示例

以下示例演示如何使用 Apache Flink DataStream API 创建应用程序。

主题

- 示例：滚动窗口 (p. 146)
- 示例：滑动窗口 (p. 152)
- 示例：写入 Amazon S3 存储桶 (p. 158)
- 教程：使用 Kinesis Data Analytics 应用程序将数据从 MSK 集群中的一个主题复制到 VPC 中的另一个主题 (p. 167)
- 示例：将 EFO 消费者与 Kinesis 数据流一起使用 (p. 171)
- 示例：写入 Kinesis Data Firehose (p. 178)
- 示例：从不同账户的 Kinesis 流中读取 (p. 188)
- 教程：在 Amazon MSK 上使用自定义信任库 (p. 194)

示例：滚动窗口

在本练习中，您将创建一个 Kinesis Data Analytics 应用程序，该应用程序使用滚动窗口聚合数据。默认情况下，聚合在 Flink 中已启用。要禁用它，请使用下面的命令：

```
sink.producer.aggregation-enabled' = 'false'
```

> Note
>
> 要为本练习设置所需的先决条件，请先完成入门指南 (DataStream API) (p. 87)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，您需要创建以下依赖资源：

- 两个 Kinesis 数据流（`ExampleInputStream`和`ExampleOutputStream`）
- 用于存储应用程序代码的 Amazon S3 存储桶的`ka-app-code-<username>`

您可以使用控制台创建 Kinesis Streams Streams 存储桶和Amazon S3 存储桶和 有关创建这些资源的说明，请参阅以下主题：

- 在 Amazon Kinesis Data Streams 开发人员指南中@@ 创建和更新数据流。将数据流命名为 **`ExampleInputStream`** 和 **`ExampleOutputStream`**。
- 如何创建 S3 存储桶？ 在 Amazon Simple Storage Serv ice 通过附加您的登录名，为 Amazon S3 存储桶指定一个全球唯一的名称，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', region_name='us-west-2'))
```

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

此示例的 Java 应用程序代码可从中获得 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：

- 该应用程序使用 Kinesis 源从源码流中读取数据。以下片段创建了 Kinesis 源代码：

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

- 添加以下导入语句：

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13 onward
```

- 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。以下代码创建运算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

                    .window(TumblingProcessingTimeWindows.of(Time.seconds(5))) //Flink
 1.13 onward
                    .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，请参阅入门指南 (DataStream API) (p. 87)教程中的先决条件 (p. 87)。
2. 使用以下命令编译应用程序：

```
mvn package -Dflink.version=1.15.3
```

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

在本部分中，您将应用程序代码上传到您在该创建相关资源 (p. 147)部分创建的 Amazon S3 存储桶。

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 Amazon S3 桶中，您的应用程序可以在其中访问。

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，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 1.15.2。
   - 将版本下拉列表保留为 Apache Flink 版本 1.15.2（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 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>/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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
6. 选择更新。

   Note

   当您选择启用 CloudWatch 日志记录时，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`

   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MyApplication页面上，选择 “运行”。保持 “不使用快照运行” 选项处于选中状态，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)。

您可以在 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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

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页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择操作，选择删除，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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

示例：滑动窗口

> **Note**
>
> 要为本练习设置所需的先决条件，请先完成入门指南 (DataStream API) (p. 87)练习。

本主题包含下列部分：

- 上传 Apache Flink 流式处理 Java 代码 (p. 155)
- 创建并运行 Kinesis Data Analytics 应用程序 (p. 155)
- 清理 Amazon 资源 (p. 157)

创建相关资源

在为本练习创建 Kinesis Data Analytics 应用程序之前，您需要创建以下依赖资源：

- 两个 Kinesis 数据流（`ExampleInputStream` 和 `ExampleOutputStream`）。
- 用于存储应用程序代码的 Amazon S3 存储桶的`ka-app-code-<username>`

您可以使用控制台创建 Kinesis Streams Streams 存储桶和Amazon S3 存储桶和 有关创建这些资源的说明，请参阅以下主题：

- 在 Amazon Kinesis Data Streams 开发人员指南中@@ 创建和更新数据流。将数据流命名为 **`ExampleInputStream`** 和 **`ExampleOutputStream`**。
- 如何创建 S3 存储桶？ 在 Amazon Simple Storage Serv ice 通过附加您的登录名，为 Amazon S3 存储桶指定一个全球唯一的名称，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

此示例的 Java 应用程序代码可从中获得 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：

- 该应用程序使用 Kinesis 源从源码流中读取数据。以下片段创建了 Kinesis 源代码：

```
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。以下代码创建运算符并将聚合的数据发送到新的 Kinesis Data Streams 接收器：
- 添加以下导入语句：

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13 onward
```

- 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。以下代码创建运算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

  .window(TumblingProcessingTimeWindows.of(Time.seconds(5))) //Flink 1.13 onward
                .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，请参阅入门指南 (DataStream API) (p. 87)教程中的先决条件 (p. 87)。
2. 使用以下命令编译应用程序：

```
mvn package -Dflink.version=1.15.3
```

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您在本创建相关资源 (p. 153)节中创建的 Amazon S3 存储桶。

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 Amazon S3 桶中，您的应用程序可以在其中访问。

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，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。
   - 将版本下拉列表保留为 Apache Flink 版本 1.15.2 (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 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>/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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
6. 选择更新。

   Note

   当您选择启用亚马逊 CloudWatch 日志记录时，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`

   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DescribeApplication或ListApplications操作检索当前应用程序版本 ID。

运行应用程序

可以通过运行应用程序、打开 Apache Flink 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

您可以在 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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 (p. 158)

删除 Kinesis Data Analytics

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页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择操作，选择删除，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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

示例：写入 Amazon S3 存储桶

在本练习中，您将创建适用于 Apache FlAmazon Kinesis Data Analytics，该数据流以 Kinesis 数据流作为源，将 Amazon S3 存储桶作为接收器。使用 simple Storage S3 控制台中验证应用程序的输出。

Note

要为本练习设置所需的先决条件，请先完成入门指南 (DataStream API) (p. 87)练习。

本主题包含下列部分：

创建相关资源

在为本练习创建适用于 Apache Flink 的 Amazon Kinesis 数据分析之前，您需要创建以下依赖资源：

- Kinesis 数据流 (`ExampleInputStream`)。
- 用于存储应用程序代码和输出的 Amazon S3 存储桶和输出（`ka-app-code-<username>`）

Note

在 Kinesis Data Analytics 上启用服务器端加密后，适用于 Apache Flink 的 Kinesis Data Analytics 无法将数据写入Amazon S3。

您可以使用控制台创建 Kinesis 直播桶和Amazon S3 存储桶和Amazon S3 存储桶 有关创建这些资源的说明，请参阅以下主题：

- 在 Amazon Kinesis Data Streams 开发人员指南中@@ 创建和更新数据流。将数据流命名为 **`ExampleInputStream`**。
- 如何创建 S3 存储桶？ 在 Amazon Simple Storage Serv ice 通过附加您的登录名，为 Amazon S3 存储桶指定一个全球唯一的名称，例如**`ka-app-code-<username>`**。在 Amazon S3 存储桶中创建两个文件夹（**`code`**和**`data`**）。

如果以下 CloudWatch 资源尚不存在，则应用程序将创建这些资源：

- 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', region_name='us-west-2'))
```

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

此示例的 Java 应用程序代码可从中获得 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：

- 该应用程序使用 Kinesis 源从源码流中读取数据。以下片段创建了 Kinesis 源代码：

```
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;
```

- 该应用程序使用 Amazon S3 S3 接收桶来写入 Amazon S3 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。以下代码对要发送到 Amazon S3 的对象进行编码：

```
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(v -> v.f0) // Logically partition the stream for each word
            .window(TumblingProcessingTimeWindows.of(Time.minutes(1)))
            .sum(1) // Count the appearances by ticker per partition
            .map(value -> value.f0 + " count: " + value.f1.toString() + "\n")
            .addSink(createS3SinkFromStaticConfig());
```

  Note

  该应用程序使用 FlinkStreamingFileSink 对象写入Amazon S3。有关 Apache 的更多信息StreamingFileSink，请参阅 StreamingFileSinkApache Flink 文档。

修改应用程序代码

在本节中，您将修改应用程序代码以将输出写入您的 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-code-<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，请参阅入门指南 (DataStream API) (p. 87)教程中的先决条件 (p. 87)。
2. 使用以下命令编译应用程序：

```
mvn package -Dflink.version=1.15.3
```

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

在本部分中，您将应用程序代码上传到您在该创建相关资源 (p. 159)部分创建的 Amazon S3 存储桶。

1. 在 Amazon S3 控制台中，选择 ka-app-code- <username>存储桶，导航到代码文件夹，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 Amazon S3 桶中，您的应用程序可以在其中访问。

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，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。
   - 将版本下拉列表保留为 Apache Flink 版本 1.15.2（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：

   - 对于 Application name (应用程序名称)，输入 **MyApplication**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。
   - 将该版本保留为 Apache Flink 版本 1.15.2（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. 选择创建应用程序。

   Note

   当您使用控制台创建适用于 Apache Flink 的 Amazon Kinesis 数据分析时，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 策略以添加权限以访问 Kinesis 数据流。

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。<username>替换为您的用户名。

```
{
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
```

```
                "s3:Abort*",
                "s3:DeleteObject*",
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-<username>",
                "arn:aws:s3:::ka-app-code-<username>/*"
            ]
          },
        {
            "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"
        },

    ]
}
```

配置应用程序

1. 在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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。

- 在 Amazon S3 对象的路径中，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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
6. 选择更新。

Note

当您选择启用 CloudWatch 日志记录时，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`

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MyApplication页面上，选择 “运行”。保持 “不使用快照运行” 选项处于选中状态，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 S3 存储桶中的数据文件夹。

几分钟后，将显示包含来自应用程序的聚合数据的对象。

Note

默认情况下，聚合在 Flink 中已启用。要禁用它，请使用下面的命令：

```
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接收器对象的设置。

Note

更改以下各节中描述的代码段后，执行以下操作以重新加载应用程序代码：

- 重复该the section called “编译应用程序代码” (p. 161)部分中的步骤以编译更新的应用程序代码。
- 重复the section called “上传 Apache Flink 流式处理 Java 代码” (p. 161)部分中的步骤，上传更新的应用程序代码。
- 在控制台的应用程序页面上，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Data 分区 (p. 165)

配置Data 分区

在本节中，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。您可以通过向流文件接收器添加存储桶分配器来完成此操作。

要自定义在 S3 存储桶中创建的文件夹名称，请执行以下操作：

1. 将以下导入语句添加到S3StreamingSinkJob.java文件开头：

```
import
 org.apache.flink.streaming.api.functions.sink.filesystem.rollingpolicies.DefaultRollingPolicy;
import
 org.apache.flink.streaming.api.functions.sink.filesystem.bucketassigners.DateTimeBucketAssigner;
```

2. 将代码中的createS3SinkFromStaticConfig()方法更新为类似于以下内容：

```
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带有自定义日期格式的，在 S3 存储桶中创建文件夹。DateTimeBucketAssigner使用当前系统时间来创建存储段名称。如果您想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，则可以创建一个实现的类BucketAssigner。您可以使用getBucketId方法实现您的自定义逻辑。

的自定义实现BucketAssigner可以使用 Co ntext 参数获取有关记录的更多信息，以确定其目标文件夹。

配置读取频率

在本节中，您将配置源流的读取频率。

默认情况下，Kinesis Streams 消费者每秒从源数据流读取五次。如果有多个客户端从流中读取数据，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 使用者的读取频率，请设置该SHARD_GETRECORDS_INTERVAL_MILLIS设置。

以下代码示例将设置设置SHARD_GETRECORDS_INTERVAL_MILLIS设置为一秒：

```
kinesisConsumerConfig.setProperty(ConsumerConfigConstants.SHARD_GETRECORDS_INTERVAL_MILLIS,
 "1000");
```

配置写入缓冲

在本节中，您将配置 sink 的写入频率和其他设置。

默认情况下，应用程序每分钟写入一次目标存储桶。您可以通过配置DefaultRollingPolicy对象来更改此间隔和其他设置。

Note

每次应用程序创建检查点时，Apache Flink 流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。

若要配置DefaultRollingPolicy对象，请执行以下操作：

1. 增加应用程序的CheckpointInterval设置。 UpdateApplication操作的以下输入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ListApplications操作检索应用程序版本。

2. 将以下导入语句添加到S3StreamingSinkJob.java文件开头：

```
import java.util.concurrent.TimeUnit;
```

3. 更新S3StreamingSinkJob.java文件中的createS3SinkFromStaticConfig方法，使其如下所示：

```
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 流式文件接收器的更多信息，请参阅 Apache Flink 文档中的行编码格式。

清理 Amazon 资源

本节包括清理您在 Amazon S3 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 (p. 167)
- 删除 Kinesis Data Streams (p. 167)

- 删除您的Amazon S3 对象和存储桶 (p. 167)
- 删除您的 IAM 资源 (p. 167)
- 删除您的 CloudWatch 资源 (p. 167)

删除 Kinesis Data Analytics

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 “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 Stream”，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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

教程：使用 Kinesis Data Analytics 应用程序将数据从 MSK 集群中的一个主题复制到 VPC 中的另一个主题

以下教程演示了如何创建带有 Amazon MSK 集群和两个主题的 Amazon VPC，以及如何创建从一个 Amazon MSK 主题读取并写入另一个主题的 Kinesis Data Analytics 应用程序。

Note

要为本练习设置所需的先决条件，请先完成入门指南 (DataStream API) (p. 87)练习。

本教程包含以下部分：

使用 Amazon MSVPC 集群创建一个 Amazon MSK 集群

要创建示例 VPC 和 Amazon MSK 集群以从 Kinesis Data Analytics 应用程序进行访问，请按照亚马逊 MSK 入门教程进行操作。

在完成本教程时，请注意以下几点：

- 在步骤 3：创建主题中，重复该`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（*`ClusterArn`*替换为 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"
}
```

- 按照教程中的步骤进行操作时，请务必在代码、命令和控制台条目中使用所选Amazon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。

此示例的 Java 应用程序代码可从中获得 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=1.15.3
```

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您在入门指南 (DataStream API) (p. 87)教程中创建的 Amazon S3 存储桶。

Note

如果您从入门教程中删除了 Amazon S3 存储桶，请再次执行该the section called “上传 Apache Flink 流式处理 Java 代码” (p. 92)步骤。

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 Amazon S3 桶中，您的应用程序可以在其中访问。

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**。
   - 对于运行时，选择 Apache Flink 版本 1.15.2。
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`*

配置应用程序

1. 在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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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

   当您使用控制台（例如 CloudWatch 日志或 Amazon VPC）指定应用程序资源时，控制台会修改您的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)。输入以下属性：

组 ID	键	值
KafkaSource	topic	AmazonKafkaTutorialTopic
KafkaSource	bootstrap.servers	############
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSource	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)。输入以下属性：

组 ID	键	值
KafkaSink	topic	AmazonKafkaTutorialTopicDestination
KafkaSink	bootstrap.servers	############
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSink	ssl.truststore.password	changeit
KafkaSink	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您的 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. 25)。
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. 要进行CloudWatch 记录，请选中 “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. 选择更新。

Note

当您选择启用 CloudWatch 日志记录时，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`

该日志流用于监控应用程序。

运行应用程序

可以通过运行应用程序、打开 Apache Flink 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

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 Amazon MSK 入门教程中的步骤 6：生成和使用数据中的步骤进行操作。

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，请参阅问题排查 (p. 397)主题中的无法访问 VPC 中的资源 (p. 402)一节。

示例：将 EFO 消费者与 Kinesis 数据流一起使用

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 (EFO) 消费者从 Kinesis 数据流中读取。如果 Kinesis 使用者使用 EFO，则 Kinesis Data Streams 服务会为其提供自己的专用带宽，而不是让使用者与其他使用者共享直播的固定带宽。

有关对 Kinesis 消费者使用 EFO 的更多信息，请参阅 FLIP-128：增强版 Kinesis 消费者风扇输出功能。

您在本示例中创建的应用程序使用Amazon Kinesis Connectoflink-connector-kinesis r () 1.15.3。

Note

要为本练习设置所需的先决条件，请先完成入门指南 (DataStream API) (p. 87)练习。

本主题包含下列部分：

- 创建相关资源 (p. 172)
- 将示例记录写入输入流 (p. 172)
- 下载并检查应用程序代码 (p. 173)

创建相关资源

在为本练习创建 Kinesis Data Analytics 应用程序之前，您需要创建以下依赖资源：

- 两个 Kinesis 数据流（`ExampleInputStream`和`ExampleOutputStream`）
- 用于存储应用程序代码的 Amazon S3 存储桶的`ka-app-code-<username>`

您可以使用控制台创建 Kinesis Streams Streams 存储桶和Amazon S3 存储桶和 有关创建这些资源的说明，请参阅以下主题：

- 在 Amazon Kinesis Data Streams 开发人员指南中@@ 创建和更新数据流。将数据流命名为 **`ExampleInputStream`** 和 **`ExampleOutputStream`**。
- 如何创建 S3 存储桶？ 在 Amazon Simple Storage Serv ice 通过附加您的登录名，为 Amazon S3 存储桶指定一个全球唯一的名称，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', region_name='us-west-2'))
```

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

此示例的 Java 应用程序代码可从中获得 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 使用者上设置以下参数来启用 EFO 使用者：
  - RECORD_PUBLISHER_TYPE：将此参数设置为 EFO，让您的应用程序使用 EFO 使用者访问 Kinesis Data Stream 数据。
  - EFO_CONSUMER_NAME：将此参数设置为在该直播的使用者中唯一的字符串值。在同一 Kinesis Data Stream 中重复使用消费者名称将导致先前使用该名称的使用者被终止。
- 以下代码示例演示如何为使用者配置属性赋值以使用 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，请参阅入门指南 (DataStream API) (p. 87)教程中的先决条件 (p. 87)。
2. 使用以下命令编译应用程序：

```
mvn package -Dflink.version=1.15.3
```

> **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

在本部分中，您将应用程序代码上传到您在该创建相关资源 (p. 172)部分创建的 Amazon S3 存储桶。

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 Amazon S3 桶中，您的应用程序可以在其中访问。

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，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 1.15.2。
   - 将版本下拉列表保留为 Apache Flink 版本 1.15.2 (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 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。

   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-efo-flink-app",
                "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream/
consumer/my-efo-flink-app:*"
            ]
        }
    ]
}
```

配置应用程序

1. 在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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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 “属性” 下，选择 “创建群组”。
5. 输入以下应用程序属性和值：

组 ID	键	值
ConsumerConfigProperties	**flink.stream.recordpublisher**	**EFO**
ConsumerConfigProperties	**flink.stream.efo.consumername**	**basic-efo-flink-app**
ConsumerConfigProperties	**INPUT_STREAM**	**ExampleInputStream**
ConsumerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ConsumerConfigProperties	**AWS_REGION**	**us-west-2**

6. 在 “属性” 下，选择 “创建群组”。
7. 输入以下应用程序属性和值：

组 ID	键	值
ProducerConfigProperties	**OUTPUT_STREAM**	**ExampleOutputStream**
ProducerConfigProperties	**AWS_REGION**	**us-west-2**
ProducerConfigProperties	**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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
10. 选择更新。

Note

当您选择启用 CloudWatch 日志记录时，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`

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 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

您可以在 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，以验证应用程序是否正常运行。

您还可以在数据流的增强扇出选项卡中查看 Kinesis Data Streams 控制台中的消费者的姓名（basic-efo-flink-app）。

清理 Amazon 资源

本节包括清理在 efo Window 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 (p. 177)
- 删除 Kinesis Data Streams (p. 177)
- 删除您的Amazon S3 对象和存储桶 (p. 177)
- 删除您的 IAM 资源 (p. 178)
- 删除您的 CloudWatch 资源 (p. 178)

删除 Kinesis Data Analytics

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页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择操作，选择删除，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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

示例：写入 Kinesis Data Firehose

在本练习中，您将创建一个 Kinesis Data Analytics 应用程序，该应用程序以 Kinesis 数据流作为源，将 Kinesis Data Firehose 传输流作为接收器。您可以使用 simple StoragAmazon S3 桶中验证应用程序的输出。

> Note
>
> 要为本练习设置所需的先决条件，请先完成入门指南 (DataStream API) (p. 87)练习。

本节包含以下步骤：

- 创建相关资源 (p. 178)
- 将示例记录写入输入流 (p. 179)
- 下载并检查 Apache Flink 流式处理 Java 代码 (p. 179)
- 编译应用程序代码 (p. 180)
- 上传 Apache Flink 流式处理 Java 代码 (p. 180)
- 创建并运行 Kinesis Data Analytics 应用程序 (p. 181)
- 清理 Amazon 资源 (p. 187)

创建相关资源

在为本练习创建适用于 Apache Flink 的 Amazon Kinesis 数据分析之前，您需要创建以下依赖资源：

- 一个 Kinesis 数据流 (`ExampleInputStream`)
- 一个 Kinesis Data Firehose 传输流，应用程序将输出写入该传输流 (`ExampleDeliveryStream`)。
- 用于存储应用程序代码的 Amazon S3 存储桶的`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：

- 在 Amazon Kinesis Data Streams 开发人员指南中@@ 创建和更新数据流。将数据流命名为 **ExampleInputStream**。
- 在 Amazon Kinesis Data Firehose 开发人员指南中创建 Amazon Kinesis Data Firehose 将传输流命名为 **ExampleDeliveryStream**。在创建 Kinesis Data Firehose 传输流时，还要创建该交付流的 S3 目标和 IAM 角色。
- 如何创建 S3 存储桶？ 在 Amazon Service 用户指南中。通过附加您的登录名，为 Amazon S3 存储桶指定一个全球唯一的名称，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', region_name='us-west-2'))
```

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

此示例的 Java 应用程序代码可从中获得 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/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：

- 该应用程序使用 Kinesis 源从源码流中读取数据。以下片段创建了 Kinesis 源代码：

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

- 该应用程序使用 Kinesis Data Firehose 接收器将数据写入传输流。以下片段创建了 Kinesis Data Firehose 水槽：

```
private static KinesisFirehoseSink<String> createFirehoseSinkFromStaticConfig() {
        Properties sinkProperties = new Properties();
        sinkProperties.setProperty(AWS_REGION, region);

        return KinesisFirehoseSink.<String>builder()
                .setFirehoseClientProperties(sinkProperties)
                .setSerializationSchema(new SimpleStringSchema())
                .setDeliveryStreamName(outputDeliveryStreamName)
                .build();
    }
```

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，请参阅入门指南 (DataStream API) (p. 87)教程中的先决条件 (p. 87)。
2. 要将 Kinesis 连接器用于以下应用程序，你需要下载、构建和安装 Apache Maven。有关更多信息，请参阅。the section called “在之前的 Apache Flink Kinesis Streams 连接器中使用 Apache Flink” (p. 323)
3. 使用以下命令编译应用程序：

```
mvn package -Dflink.version=1.15.3
```

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您在本创建相关资源 (p. 178)节中创建的 Amazon S3 存储桶。

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 Uplo ad。

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 Amazon S3 桶中，您的应用程序可以在其中访问。

创建并运行 Kinesis Data Analytics 应用程序

您可以使用控制台或通过创建并运行 Kinesis Data Analytics 应用程序Amazon CLI。

> Note
>
> 当您使用控制台创建应用程序时，会为您创建Amazon Identity and Access Management (IAM) 和 A CloudWatch mazon Logs 资源。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，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
     >
     > 适用于 Apache Flink 的 Kinesis Data Analytics 使用 Apache Flink 版本 1.15.2。

   - 将版本下拉列表保留为 Apache Flink 版本 1.15.2（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. 选择创建应用程序。

> Note
>
> 当您使用控制台创建应用程序时，您可以选择为应用程序创建 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 数据流和 Kinesis Data Firehose 传输流的权限。

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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
6. 选择更新。

> **Note**
>
> 当您选择启用 CloudWatch 日志记录时，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 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

停止应用程序

在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. 186)一节中所述的 Amazon CLI。如果文件名或存储段未更改，则当您在 “配置” 页面上选择 “更新” 时，不会重新加载应用程序代码。

创建并运行应用程序 (Amazon CLI)

在本节中，您将使用创建和运行 Kinesis Data Analytics 应用程序。Amazon CLI

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（您在下一节中创建）。因此，当 Kinesis Data Analytics 担任该角色时，该服务具有从源流读取和写入接收流的必要权限。

使用以下代码创建 `KAReadSourceStreamWriteSinkStream` 权限策略。将*###*替换为用于创建 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": "WriteDeliveryStream",
            "Effect": "Allow",
            "Action": "firehose:*",
            "Resource": "arn:aws:firehose:us-west-2:012345678901:deliverystream/
ExampleDeliveryStream"
        }
    ]
}
```

有关创建权限策略的 step-by-step 说明，请参阅 IAM 用户指南中的教程：创建并附加您的第一个客户托管策略。

> **Note**
>
> 要访问其他亚马逊服务，您可以使用Amazon SDK for Java。Kinesis Data Analytics 会自动将软件开发工具包所需的证书设置为与您的应用程序关联的服务执行 IAM 角色的证书。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 应用程序可以假定该角色来读取源流并写入接收流。

如果 Kinesis Data Analytics 没有权限，则无法访问您的直播。您可以通过 IAM 角色授予这些权限。每个 IAM 角色都附加了两个策略。信任策略授予 Kinesis Data Analytics 权限以代入该角色。权限策略决定了 Kinesis Data Analytics 在担任该角色后可以做什么。

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 “选择可信身份类型” 下，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 Attach permissions policies 页面上，选择 Next: Review。在创建角色后，您可以附加权限策略。
5. 在创建角色页面上**`KA-stream-rw-role`**，输入角色名称。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 Analytics 扮演这个角色，既可以从 Kinesis 数据流（源）读取数据，也可以将输出写入另一个 Kinesis 数据流。因此，您附加在上一步（the section called “创建权限策略” (p. 184)）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。

有关创建角色的 step-by-step 说明，请参阅 IAM 用户指南中的创建 IAM 角色（控制台）。

创建 Kinesis Data Analytics

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. 178)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_15",
    "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 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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 日志选项

您可以使用Amazon CLI向您的应用程序添加 Amazon CloudWatch 日志流。有关在应用程序中使用 CloudWatch Logs 的信息，请参阅the section called “设置日志记录” (p. 269)。

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。

要使用Amazon CLI，请从 Amazon S3 存储桶中删除之前的代码包，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. 178)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

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页面中，选择 “删除 Kinesis Stream”，然后确认删除。

删除您的 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页面中，选择删除传输流，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- 存储桶。 <username>
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 传输流的目标创建了 Amazon S3 存储桶，请同时删除该存储桶。

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>策略。
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，请同时删除该策略。
7. 在导航栏中，选择 Roles（角色）。
8. 选择 k inesis-analyticsMyApplication- <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 交付流创建了新角色，请同时删除该角色。

删除您的 CloudWatch 资源

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

示例：从不同账户的 Kinesis 流中读取

此示例演示如何创建一个 Amazon Kinesis Data Analytics 应用程序，该应用程序从不同账户的 Kinesis 直播中读取数据。在此示例中，您将为源 Kinesis 直播使用一个帐户，为 Kinesis Data Analytics 应用程序和接收器 Kinesis 直播使用第二个帐户。

本主题包含下列部分：

先决条件

- 在本教程中，您将修改入门示例以读取其他账户中的 Kinesis 直播中的数据。在继续之前，请完成入门指南 (DataStream API) (p. 87)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，使其包含两个配置文件，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一起使用的更多信息Amazon CLI，请参阅Amazon Command Line Interface文档中的命名配置文件。

创建源 Kinesis 流

在本节中，您将在源账户中创建 Kinesis 直播。

输入以下命令以创建应用程序将用于输入的 Kinesis 流。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

要允许跨Amazon账户访问对象，您需要在源账户中创建 IAM 角色和策略。然后，您修改 sink 账户中的 IAM 策略。有关创建 IAM 角色和策略的信息，请参阅《Amazon Identity and Access Management用户指南》中的以下主题：

- 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，控制台会创建一个名为的策略和一个名kinesis-analytics-service-*<application name>*-*<application region>*为的角色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 “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: Review（下一步: 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. KA-Source-Stream-Role在 IAM 控制台中打开。
    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 KinesisStreamsSink<String> createSinkFromStaticConfig() {
        Properties outputProperties = new Properties();
        outputProperties.setProperty(AWSConfigConstants.AWS_REGION, region);

        return KinesisStreamsSink.<String>builder()
                .setKinesisClientProperties(outputProperties)
                .setSerializationSchema(new SimpleStringSchema())
                .setStreamName(outputProperties.getProperty("OUTPUT_STREAM",
 "ExampleOutputStream"))
                .setPartitionKeyGenerator(element -> String.valueOf(element.hashCode()))
                .build();
    }

    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=1.15.3
```

2. 从您的Amazon Simple Storage Service (Amazon S3) 存储桶删除之前的 JAR 文件，然后将新aws-kinesis-analytics-java-apps-1.0.jar文件上传到 S3 存储桶。
3. 在 Kinesis Data Analytics 控制台的应用程序页面中，选择配置、更新以重新加载应用程序 JAR 文件。
4. 运行 stock.py 脚本以将数据发送到源流。

```
python stock.py
```

现在，该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：在 Amazon MSK 上使用自定义信任库

以下教程演示了如何安全地连接（传输中加密）到 Kafka 集群，该集群使用自定义、私有甚至自托管的证书颁发机构 (CA) 颁发的服务器证书。

为了通过 TLS 将任何 Kafka 客户端安全地连接到 Kafka 集群，Kafka 客户端（例如示例 Flink 应用程序）必须信任 Kafka 集群的服务器证书（从发行 CA 到根级 CA）提供的完整信任链。作为自定义 Truststore 的示例，我们将使用启用双向 TLS (MTLS) 身份验证的 Amazon MSK 集群。这意味着 MSK 集群节点使用由 Certificate Manager 私有Amazon证书颁发机构 (ACM Private CA) 颁发的服务器证书，该证书是您的账户和区域的私有证书，因此执行 Flink 应用程序的 Java 虚拟机 (JVM) 的默认信任库不受信任。

Note

- 密钥库用于存储应用程序应向服务器或客户端出示的私钥和身份证书以供验证。
- Truststore 用于存储来自认证机构 (CA) 的证书，这些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进行 Kinesis Data Analytics 应用程序与其他 Apache Kafka 源之间的交互，例如：

- 托管在Amazon（A mazon EC2 或亚马逊 EKS）中的自定义 Apache Kafka 集群
- 托管在 Confluent Kafka 集群Amazon
- 通过Amazon Direct Connect或 VPN 访问的本地 Kafka 集群

您的应用程序将使用自定义序列化和反序列化架构来替代加载自定义信任库open的方法。这使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：

```
public static void initializeKafkaTruststore() {
```

```
    ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
    URL inputUrl = classLoader.getResource("kafka.client.truststore.jks");
    File dest = new File("/tmp/kafka.client.truststore.jks");

    try {
        FileUtils.copyURLToFile(inputUrl, dest);
    } catch (Exception ex) {
        throw new FlinkRuntimeException("Failed to initialize Kakfa truststore", ex);
    }
}
```

Note

Apache Flink 要求信任库采用 。

Note

要设置本练习所需的先决条件，请先完成入门指南 (DataStream API) (p. 87)练习。

本教程包含以下部分：

- 使用亚马逊 MSK 集群创建 VPC (p. 195)
- 创建自定义信任库并将其应用于您的集群 (p. 196)
- 创建应用程序代码 (p. 196)
- 上传 Apache Flink 流式处理 Java 代码 (p. 196)
- 创建 应用程序 (p. 197)
- 配置应用程序 (p. 197)
- 运行应用程序 (p. 198)
- 测试应用程序 (p. 198)

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 应用程序进行访问，请按照亚马逊 MSK 入门教程进行操作。

完成本教程时，还要执行以下操作：

- 在步骤 3：创建主题中，重复该`kafka-topics.sh --create`命令以创建名为的目标主题`AmazonKafkaTutorialTopicDestination`：

```
bin/kafka-topics.sh --create --bootstrap-server ZooKeeperConnectionString --replication-
factor 3 --partitions 1 --topic AWSKafkaTutorialTopicDestination
```

Note

如果`kafka-topics.sh`命令返回`ZooKeeperClientTimeoutException`，请验证 Kafka 集群的安全组是否具有允许来自客户端实例私有 IP 地址的所有流量的入站规则。

- 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（*ClusterArn*替换为 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"
```

```
}
```

- 在按照本教程和先决条件教程中的步骤进行操作时，请务必在代码、命令和控制台条目中使用所选Amazon区域。

创建自定义信任库并将其应用于您的集群

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，并将其应用于您的 MSK 集群。

要创建和应用您的自定义信任库，请按照适用于 A pache Kafka 的 Amazon Managed Streaming Streaming 开发者指南中的客户端身份验证教程进行操作。

创建应用程序代码

在本节中，您将下载并编译应用程序 JAR 文件。

此示例的 Java 应用程序代码可从中获得 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。你可以检查代码，熟悉适用于 Apache FlAmazon Kinesis Data Analytics 代码的结构。
4. 使用命令行 Maven 工具或首选开发环境创建 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=1.15.3
```

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您入门指南 (DataStream API) (p. 87)在本教程中创建的 Amazon S3 存储桶。

> **Note**
>
> 如果您从入门教程中删除了 Amazon S3 存储桶，请再次执行该the section called “上传 Apache Flink 流式处理 Java 代码” (p. 92)步骤。

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 flink-app-1.0-SNAPSHOT.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 桶中，您的应用程序可以在其中访问。

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**。
   - 对于运行时，选择 Apache Flink 版本 1.15.2。
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. 选择创建应用程序。

   Note

   当您使用控制台创建适用于 Apache Flink 的 Amazon Kinesis 数据分析时，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*

配置应用程序

1. 在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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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

   当您使用控制台指定应用程序资源（例如日志或 VPC）时，控制台会修改您的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)。输入以下属性：

组 ID	键	值
KafkaSource	topic	AmazonKafkaTutorialTopic
KafkaSource	bootstrap.servers	############
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSource	ssl.truststore.password	changeit

Note

默认证书的 ssl.truststore.pas sword 是 “更改它”，如果您使用的是默认证书，则无需更改此值。

再次选择 Add Group (添加组)。输入以下属性：

组 ID	键	值
KafkaSink	topic	AmazonKafkaTutorialTopicDestination
KafkaSink	bootstrap.servers	############
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSink	ssl.truststore.password	changeit
KafkaSink	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您的 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. 25)。

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. 要进行CloudWatch 记录，请选中 “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. 选择更新。

Note

当您选择启用 CloudWatch 日志记录时，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`

该日志流用于监控应用程序。

运行应用程序

可以通过运行应用程序、打开 Apache Flink 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

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。您可以通过向源主题写入记录并从目标主题读取记录来验证应用程序是否正常运行。

要写入和读取主题中的记录，请按照 Amazon MSK 入门教程中的步骤 6：生成和使用数据中的步骤进行操作。

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，请参阅问题排查 (p. 397)主题中的无法访问 VPC 中的资源 (p. 402)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. 199)
- 示例：在 Python 中创建滑动窗口 (p. 206)
- 示例：使用 Python 向Amazon S3 发送流媒体数据 (p. 212)

示例：在 Python 中创建翻滚窗口

在本练习中，您将创建一个 Python Kinesis Data Analytics 应用程序，该应用程序使用滚动窗口聚合数据。

> Note
>
> 要为本练习设置所需的先决条件，请先完成入门指南 (Python) (p. 113)练习。

本主题包含下列部分：

- 创建相关资源 (p. 199)
- 将示例记录写入输入流 (p. 200)
- 下载并检查应用程序代码 (p. 200)
- 压缩并上传 Apache Flink 直播 Python 代码 (p. 201)
- 创建并运行 Kinesis Data Analytics 应用程序 (p. 202)
- 清理 Amazon 资源 (p. 204)

创建相关资源

在为本练习创建 Kinesis Data Analytics 应用程序之前，您需要创建以下依赖资源：

- 两个 Kinesis 数据流（`ExampleInputStream`和`ExampleOutputStream`）
- 用于存储应用程序代码的 Amazon S3 存储桶的`ka-app-code-<username>`

您可以使用控制台创建 Kinesis Streams Streams 存储桶和Amazon S3 存储桶和 有关创建这些资源的说明，请参阅以下主题：

- 在 Amazon Kinesis Data Streams 开发人员指南中@@ 创建和更新数据流。将数据流命名为 **`ExampleInputStream`** 和 **`ExampleOutputStream`**。
- 如何创建 S3 存储桶？ 在 Amazon Service 用户指南中。通过附加您的登录名，为 Amazon S3 存储桶指定一个全球唯一的名称，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. 您必须将您的配置Amazon CLI为使用您的账户凭证和默认区域。若要配置您的Amazon CLI，请输入以下内容：

```
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', region_name='us-west-2'))
```

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

此示例的 Python 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. 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：

- 该应用程序使用 Kinesis 表源从源数据流中读取数据。以下代码段调用该create_table函数来创建 Kinesis 表源：

```
table_env.execute_sql(
        create_input_table(input_table_name, input_stream, input_region, stream_initpos)
    )
```

该create_table函数使用 SQL 命令创建由流式传输源支持的表：

```
def create_input_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}',
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(table_name, stream_name, region, stream_initpos)
```

- 该应用程序使用Tumble运算符聚合指定的 trumbling 窗口内的记录，并将聚合的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.min as price, to_string(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-1.15.2.jar。

压缩并上传 Apache Flink 直播 Python 代码

在本部分中，您将应用程序代码上传到您在该创建相关资源 (p. 199)部分创建的 Amazon S3 存储桶。

1. 使用您首选的压缩应用程序压缩tumbling-windows.py和flink-sql-connector-kinesis-1.15.2.jar文件。命名档案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 Amazon S3 桶中，您的应用程序可以在其中访问。

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，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 1.15.2。
   - 将版本下拉列表保留为 Apache Flink 版本 1.15.2（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`

配置应用程序

1. 在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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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. 在 “属性” 下，选择 “添加群组”。
5. 输入以下信息:

组 ID	键	值
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

选择保存。

6. 在 “属性” 下，再次选择 “添加群组”。
7. 输入以下信息:

组 ID	键	值
producer.config.0	**output.stream.name**	**ExampleOutputStream**
producer.config.0	**aws.region**	**us-west-2**
producer.config.0	**shard.count**	**1**

8. 在 “属性” 下，再次选择 “添加群组”。对于群组 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，请参阅指定您的代码文件 (p. 23)：
9. 输入以下信息:

组 ID	键	值
kinesis.analytics.flink.run.options	**python**	**tumbling-windows.py**
kinesis.analytics.flink.run.options	**jarfile**	**flink-sql-connector-kinesis-1.15.2.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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
12. 选择更新。

Note

当您选择启用 CloudWatch 日志记录时，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`

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>/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

可以通过运行应用程序、打开 Apache Flink 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

您可以在 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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 (p. 205)

删除 Kinesis Data Analytics

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页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择操作，选择删除，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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

示例：在 Python 中创建滑动窗口

Note

要为本练习设置所需的先决条件，请先完成入门指南 (Python) (p. 113)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，您需要创建以下依赖资源：

- 两个 Kinesis 数据流（`ExampleInputStream`和`ExampleOutputStream`）
- 用于存储应用程序代码的 Amazon S3 存储桶的`ka-app-code-<username>`

您可以使用控制台创建 Kinesis Streams Streams 存储桶和Amazon S3 存储桶和 有关创建这些资源的说明，请参阅以下主题：

- 在 Amazon Kinesis Data Streams 开发人员指南中@@ 创建和更新数据流。将数据流命名为 **`ExampleInputStream`** 和 **`ExampleOutputStream`**。
- 如何创建 S3 存储桶？ 在 Amazon Service 用户指南中。通过附加您的登录名，为 Amazon S3 存储桶指定一个全球唯一的名称，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. 您必须将您的配置Amazon CLI为使用您的账户凭证和默认区域。若要配置您的Amazon CLI，请输入以下内容：

```
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', region_name='us-west-2'))
```

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

此示例的 Python 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. 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：

- 该应用程序使用 Kinesis 表源从源数据流中读取数据。以下代码段调用该create_input_table函数来创建 Kinesis 表源：

```
table_env.execute_sql(
        create_input_table(input_table_name, input_stream, input_region, stream_initpos)
    )
```

该create_input_table函数使用 SQL 命令创建由流式传输源支持的表：

```
def create_input_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}',
              '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，并将聚合的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, to_string(ten_second_window.end) as
 event_time")
    )
```

- 该应用程序使用 flink-sql-connector-kinesis-1.15.2.jar 文件中的 Kinesis Flink 连接器。

压缩并上传 Apache Flink 直播 Python 代码

在本部分中，您将应用程序代码上传到您在该创建相关资源 (p. 206)部分创建的 Amazon S3 存储桶。

本节介绍如何打包您的 Python 应用程序。

1. 使用您首选的压缩应用程序压缩sliding-windows.py和flink-sql-connector-kinesis-1.15.2.jar文件。命名档案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 Amazon S3 桶中，您的应用程序可以在其中访问。

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，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 1.15.2。

- 将版本下拉列表保留为 Apache Flink 版本 1.15.2（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*

配置应用程序

1. 在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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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. 在“属性”下，选择“添加群组”。
5. 输入以下应用程序属性和值：

组 ID	键	值
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

选择保存。

6. 在“属性”下，再次选择“添加群组”。
7. 输入以下应用程序属性和值：

组 ID	键	值
producer.config.0	**output.stream.name**	**ExampleOutputStream**
producer.config.0	**aws.region**	**us-west-2**
producer.config.0	**shard.count**	**1**

8. 在“属性”下，再次选择“添加群组”。对于群组 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，请参阅指定您的代码文件 (p. 23)：
9. 输入以下应用程序属性和值：

组 ID	键	值
kinesis.analytics.flink.run.options	**python**	**sliding-windows.py**
kinesis.analytics.flink.run.options	**jarfile**	**flink-sql-connector-kinesis_1.15.2.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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
12. 选择更新。

Note

当您选择启用 CloudWatch 日志记录时，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`

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>/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

可以通过运行应用程序、打开 Apache Flink 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

您可以在 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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 (p. 211)
- 删除 Kinesis Data Streams (p. 212)
- 删除您的Amazon S3 对象和存储桶 (p. 212)
- 删除您的 IAM 资源 (p. 212)
- 删除您的 CloudWatch 资源 (p. 212)

删除 Kinesis Data Analytics

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页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择操作，选择删除，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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

示例：使用 Python 向Amazon S3 发送流媒体数据

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，请先完成入门指南 (Python) (p. 113)练习。

本主题包含下列部分：

- 创建相关资源 (p. 213)
- 将示例记录写入输入流 (p. 213)
- 下载并检查应用程序代码 (p. 214)
- 压缩并上传 Apache Flink 直播 Python 代码 (p. 215)

- 创建并运行 Kinesis Data Analytics 应用程序 (p. 215)
- 清理 Amazon 资源 (p. 218)

创建相关资源

在为本练习创建 Kinesis Data Analytics 应用程序之前，您需要创建以下依赖资源：

- 一个 Kinesis 数据流 (`ExampleInputStream`)
- 用于存储应用程序代码和输出的 Amazon S3 存储桶 (`ka-app-code-<username>`)

> Note
>
> 在 Kinesis Data Analytics 上启用服务器端加密后，适用于 Apache Flink 的 Kinesis Data Analytics 无法将数据写入Amazon S3。

您可以使用控制台创建 Kinesis 直播桶和Amazon S3 存储桶和Amazon S3 存储桶 有关创建这些资源的说明，请参阅以下主题：

- 在 Amazon Kinesis Data Streams 开发人员指南中@@ 创建和更新数据流。将数据流命名为 **`ExampleInputStream`**。
- 如何创建 S3 存储桶？ 在 Amazon Service 用户指南中。通过附加您的登录名，为 Amazon S3 存储桶指定一个全球唯一的名称，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. 您必须将您的配置Amazon CLI为使用您的账户凭证和默认区域。若要配置您的Amazon CLI，请输入以下内容：

```
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', region_name='us-west-2'))
```

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

此示例的 Python 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. 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 streaming-file-sink.py 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 Kinesis 表源从源数据流中读取数据。以下代码段调用该create_source_table函数来创建 Kinesis 表源：

```
table_env.execute_sql(
        create_source_table(input_table_name, input_stream, input_region, stream_initpos)
    )
```

该create_source_table函数使用 SQL 命令创建由流式传输源支持的表

```
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', region_name='us-west-2'))
```

- 此应用程序使用`filesystem`连接器将记录发送到 Amazon S3 存储桶：

```
def create_sink_table(table_name, bucket_name):
    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time VARCHAR(64)
              )
              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 flink-sql-connector-kinesis-1.15.2.jar 文件中的 Kinesis Flink 连接器。

压缩并上传 Apache Flink 直播 Python 代码

在本部分中，您将应用程序代码上传到您在该创建相关资源 (p. 213)部分创建的 Amazon S3 存储桶。

1. 使用您首选的压缩应用程序压缩`streaming-file-sink.py`和 flink-sql-connector-kinesis-1.15.2.jar 文件。命名档案`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 Amazon S3 桶中，您的应用程序可以在其中访问。

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，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 1.15.2。

- 将版本下拉列表保留为 Apache Flink 版本 1.15.2（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*

配置应用程序

1. 在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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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. 在 “属性” 下，选择 “添加群组”。
5. 输入以下应用程序属性和值：

组 ID	键	值
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

选择保存。

6. 在 “属性” 下，再次选择 “添加群组”。对于群组 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，请参阅指定您的代码文件 (p. 23)：
7. 输入以下应用程序属性和值：

组 ID	键	值
kinesis.analytics.flink.run.options	**python**	**streaming-file-sink.py**
kinesis.analytics.flink.run.options	**jarfile**	**S3Sink/lib/flink-sql-connector-kinesis-1.15.2.jar**

8. 在 “属性” 下，再次选择 “添加群组”。对于群组 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，请参阅指定您的代码文件 (p. 23)：
9. 输入以下应用程序属性和值：（将存储***###***替换为您的 Amazon S3 存储桶的实际名称。）

组 ID	键	值
sink.config.0	**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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
12. 选择更新。

Note

当您选择启用 CloudWatch 日志记录时，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`

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>/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-code-<username>",
                "arn:aws:s3:::ka-app-code-<username>/*"
            ]
        }
    ]
}
```

运行应用程序

可以通过运行应用程序、打开 Apache Flink 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

您可以在 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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 (p. 219)
- 删除 Kinesis Data Streams (p. 219)
- 删除您的Amazon S3 对象和存储桶 (p. 219)
- 删除您的 IAM 资源 (p. 219)
- 删除您的 CloudWatch 资源 (p. 219)

删除 Kinesis Data Analytics

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页面中，选择 “删除 Kinesis Stream”，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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

Scala 示例

以下示例演示如何使用 Simpache Flink 创建应用程序。

主题

示例：在 Scala 中创建翻滚窗口

Note

从 1.15 版本开始，Flink 是免费的 Scala。应用程序现在可以使用任何 Scala 版本的 Java API。Flink 仍然在一些关键组件中使用 Scala，但不会将 Scala 暴露给用户代码类加载器。因此，用户需要将 Scala 依赖项添加到他们的 jar 存档中。
有关 Flink 1.15 中 Scala 变更的更多信息，请参阅 One Feifteen 中的 Scala Free。

在本练习中，您将创建一个使用 Scala 3.2.0 和 Flink 的 Java DataStream API 的简单流媒体应用程序。该应用程序从 Kinesis 流读取数据，使用滑动窗口聚合数据，并将结果写入输出 Kinesis 流。

Note

要设置本练习所需的先决条件，请先完成入门 (Scala) 练习。

本主题包含下列部分：

下载并检查应用程序代码

此示例的 Python 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. 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/scala/TumblingWindow目录。

请注意有关应用程序代码的以下信息：

- build.sbt文件包含有关应用程序配置和依赖关系的信息，包括 Kinesis Data Analytics 库。
- 该BasicStreamingJob.scala文件包含定义应用程序功能的主要方法。
- 该应用程序使用 Kinesis 源从源码流中读取数据。以下片段创建了 Kinesis 源代码：

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

该应用程序还使用 Kinesis 接收器来写入结果流。以下片段创建了 Kinesis 水槽：

```
private def createSink: KinesisStreamsSink[String] = {
```

```
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val outputProperties = applicationProperties.get("ProducerConfigProperties")

  KinesisStreamsSink.builder[String]
    .setKinesisClientProperties(outputProperties)
    .setSerializationSchema(new SimpleStringSchema)
    .setStreamName(outputProperties.getProperty(streamNameKey, defaultOutputStreamName))
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))
    .build
}
```

- 该应用程序使用窗口运算符在 5 秒钟的滚动窗口内查找每个股票代码的值数量。以下代码创建运算符并将聚合的数据发送到新的 Kinesis Data Streams 接收器：

```
environment.addSource(createSource)
  .map { value =>
    val jsonNode = jsonParser.readValue(value, classOf[JsonNode])
    new Tuple2[String, Int](jsonNode.get("ticker").toString, 1)
  }
  .returns(Types.TUPLE(Types.STRING, Types.INT))
  .keyBy(v => v.f0) // Logically partition the stream for each ticker
  .window(TumblingProcessingTimeWindows.of(Time.seconds(10)))
  .sum(1) // Sum the number of tickers per partition
  .map { value => value.f0 + "," + value.f1.toString + "\n" }
  .sinkTo(createSink)
```

- 应用程序创建源和接收连接器以使用 StreamExecutionEnvironment 对象访问外部资源。
- 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。读取运行时应用程序的属性以配置连接器。有关运行时属性的更多信息，请参阅运行时属性。

编译并上传应用程序代码

在本节中，您将编译应用程序代码并上传到 Amazon S3 存储桶。

编译应用程序代码

使用 SBT 编译工具为应用程序构建 Scala 代码。要安装 SBT，请参阅使用 cs 设置安装 sbt。您还需要安装 Java 开发工具包 (JDK)。请参阅完成练习的先决条件。

1. 要使用您的应用程序代码，您将其编译和打包成 JAR 文件。你可以用 SBT 编译和打包你的代码：

```
sbt assembly
```

2. 如果应用程序成功编译，则创建以下文件：

```
target/scala-3.2.0/tumbling-window-scala-1.0.jar
```

上传 Apache Flink 直播 Scala 代码

在此部分中，您将创建一个 Simple Storage S3 存储桶并上传应用程序代码。

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择创建存储桶
3. ka-app-code-<username>在存储段名称字段中输入。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。选择下一步。
4. 在配置选项中，保持设置不变，然后选择下一步。

5. 在设置权限中，保持设置不变，然后选择下一步。
6. 选择创建桶。
7. 选择ka-app-code-<username>存储桶然后选择 Upload（上载）。
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 tumbling-window-scala-1.0.jar 文件。
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 桶中，您的应用程序可以在其中访问。

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 Scala test app**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。
   - 将该版本保留为 Apache Flink 版本 1.15.2（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*

配置应用程序

可以按照以下步骤配置应用程序。

配置应用程序

1. 在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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，输入**tumbling-window-scala-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 “属性” 下，选择 “添加群组”。
5. 输入以下信息:

组 ID	键	值
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

选择保存。

6. 在 “属性” 下，再次选择 “添加群组”。
7. 输入以下信息:

组 ID	键	值
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
10. 选择更新。

Note

当您选择启用亚马逊 CloudWatch 日志记录时，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`

编辑 IAM 策略

编辑 IAM 策略以添加权限以访问 Amazon S3 存储桶的权限。

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/tumbling-window-scala-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"
        }
    ]
}
```

运行应用程序

可以通过运行应用程序、打开 Apache Flink 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

停止应用程序

要停止应用程序，请在MyApplication页面上选择停止。确认该操作。

创建并运行应用程序 (CLI)

在本节中，您将使用创建和运行 Kinesis Data Analytics 应用程序。Amazon Command Line Interface使用 kinesisanalyticsv2Amazon CLI 命令创建 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，您创建一个包含两个语句的权限策略：一个语句授予对源流进行读取操作的权限，另一个语句授予对接收流进行写入操作的权限。然后，您将策略附加到 IAM 角色（您在下一节中创建）。因此，当 Kinesis Data Analytics 担任该角色时，该服务具有从源流读取和写入接收流的必要权限。

使用以下代码创建 KAReadSourceStreamWriteSinkStream 权限策略。**username**替换为用于创建 Amazon S3 存储桶以存储应用程序代码的用户名。将亚马逊资源名称 (ARN) 中的账户 ID 替换**(012345678901)**为您的账户 ID。**KA-stream-rw-role**服务执行角色应根据客户的特定角色量身定制。

```
{
    "ApplicationName": "tumbling_window",
    "ApplicationDescription": "Scala tumbling window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/KA-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "tumbling-window-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
   ]
```

```
}
```

有关创建权限策略的 step-by-step 说明，请参阅 IAM 用户指南中的教程：创建并附加您的第一个客户托管策略。

创建 IAM 角色

在本节中，您将创建一个 IAM 角色，Kinesis Data Analytics 应用程序可以假定该角色来读取源流并写入接收流。

未经许可，Kinesis Data Analytics 无法访问您的直播。您可以通过 IAM 角色授予这些权限。每个 IAM 角色都附加了两个策略。信任策略授予 Kinesis Data Analytics 代入角色，权限策略决定了 Kinesis Data Analytics a

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，选择角色，然后选择创建角色。
3. 在 “选择可信身份类型” 下，选择 “Amazon服务”
4. 在 Choose the service that will use this role (选择将使用此角色的服务) 下，选择 Kinesis。
5. 在选择您的使用案例下，选择 Kinesis Analytics。
6. 选择Next: Permissions（下一步: 权限）。
7. 在 Attach permissions policies 页面上，选择 Next: Review。在创建角色后，您可以附加权限策略。
8. 在创建角色页面上**`KA-stream-rw-role`**，输入角色名称。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，更新角色的信任和权限策略
9. 将权限策略附加到角色。

   > **Note**
   >
   > 在本练习中，Kinesis Data Analytics 扮演这个角色，既可以从 Kinesis 数据流（源）读取数据，也可以将输出写入另一个 Kinesis 数据流。因此，您可以附加您在上一步中创建的策略，即创建权限策略。

   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。

有关创建角色的 step-by-step 说明，请参阅 IAM 用户指南中的创建 IAM 角色（控制台）。

创建应用程序

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（用户名）替换为您在上一节中选择的后缀。将服务执行角色中的示例账户 ID (012345678901) 替换为您的账户 ID。`ServiceExecutionRole`应包括您在上一节中创建的 IAM 用户角色。

```
"ApplicationName": "tumbling_window",
```

```
    "ApplicationDescription": "Scala getting started application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/KA-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "tumbling-window-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-group:MyApplication:log-
stream:kinesis-analytics-log-stream"
      }
   ]
}
```

CreateApplication使用以下请求执行以创建应用程序：

```
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": "tumbling_window",
    "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 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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": "tumbling_window"
}
```

2. 使用上述请求执行StopApplication操作以停止应用程序：

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

应用程序现已停止。

添加 CloudWatch 日志选项

您可以使用Amazon CLI向您的应用程序添加 Amazon CloudWatch 日志流。有关在应用程序中使用 CloudWatch 日志的信息，请参阅设置应用程序日志。

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。在此示例中，您将更改源存储桶和目标直播的区域。

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": "tumbling_window",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
```

```
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
                }
            }
        ]
    }
  }
}
```

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

当您需要使用新版本的代码包更新应用程序代码时，可以使用 UpdateApplicationCLI 操作。

> Note
>
> 要加载具有相同文件名的新版本的应用程序代码，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。

要使用Amazon CLI，请从 Amazon S3 存储桶中删除您之前的代码包，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。<username>使用您在部分中选择的后缀更新存储创建相关资源 (p. 123)段名称后缀 ()。

```
{
    "ApplicationName": "tumbling_window",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "tumbling-window-scala-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

清理 Amazon 资源

本节包括清理在 tumbling Window 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 (p. 230)
- 删除 Kinesis Data Streams (p. 230)
- 删除您的Amazon S3 对象和存储桶 (p. 230)
- 删除您的 IAM 资源 (p. 230)

- 删除您的 CloudWatch 资源 (p. 230)

删除 Kinesis Data Analytics

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页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择操作，选择删除，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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

示例：在 Scala 中创建滑动窗口

Note

从 1.15 版本开始，Flink 是免费的 Scala。应用程序现在可以使用任何 Scala 版本的 Java API。Flink 仍然在一些关键组件中使用 Scala，但不会将 Scala 暴露给用户代码类加载器。因此，用户需要将 Scala 依赖项添加到他们的 jar 存档中。

有关 Flink 1.15 中 Scala 变更的更多信息，请参阅 One Feifteen 中的 Scala Free。

在本练习中，您将创建一个使用 Scala 3.2.0 和 Flink 的 Java DataStream API 的简单流媒体应用程序。该应用程序从 Kinesis 流读取数据，使用滑动窗口聚合数据，并将结果写入输出 Kinesis 流。

Note

要设置本练习所需的先决条件，请先完成入门 (Scala) 练习。

本主题包含下列部分：

下载并检查应用程序代码

此示例的 Python 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. 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/scala/SlidingWindow目录。

请注意有关应用程序代码的以下信息：

- build.sbt文件包含有关应用程序配置和依赖关系的信息，包括 Kinesis Data Analytics 库。
- 该BasicStreamingJob.scala文件包含定义应用程序功能的主要方法。
- 该应用程序使用 Kinesis 源从源码流中读取数据。以下片段创建了 Kinesis 源代码：

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

该应用程序还使用 Kinesis 接收器来写入结果流。以下片段创建了 Kinesis 水槽：

```
private def createSink: KinesisStreamsSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val outputProperties = applicationProperties.get("ProducerConfigProperties")

  KinesisStreamsSink.builder[String]
    .setKinesisClientProperties(outputProperties)
    .setSerializationSchema(new SimpleStringSchema)
    .setStreamName(outputProperties.getProperty(streamNameKey, defaultOutputStreamName))
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))
    .build
```

```
}
```

- 该应用程序使用窗口运算符在 10 秒窗口内查找每个股票品种的值数量，该窗口滑动 5 秒。以下代码创建运算符并将聚合的数据发送到新的 Kinesis Data Streams 接收器：

```
environment.addSource(createSource)
    .map { value =>
      val jsonNode = jsonParser.readValue(value, classOf[JsonNode])
      new Tuple2[String, Double](jsonNode.get("ticker").toString,
 jsonNode.get("price").asDouble)
    }
    .returns(Types.TUPLE(Types.STRING, Types.DOUBLE))
    .keyBy(v => v.f0) // Logically partition the stream for each word
    .window(SlidingProcessingTimeWindows.of(Time.seconds(10), Time.seconds(5)))
    .min(1) // Calculate minimum price per ticker over the window
    .map { value => value.f0 + String.format(",%.2f", value.f1) + "\n" }
    .sinkTo(createSink)
```

- 应用程序创建源和接收连接器以使用 StreamExecutionEnvironment 对象访问外部资源。
- 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。读取运行时应用程序的属性以配置连接器。有关运行时属性的更多信息，请参阅运行时属性。

编译并上传应用程序代码

在本节中，您将编译应用程序代码并上传到 Amazon S3 存储桶。

编译应用程序代码

使用 SBT 编译工具为应用程序构建 Scala 代码。要安装 SBT，请参阅使用 cs 设置安装 sbt。您还需要安装 Java 开发工具包 (JDK)。请参阅完成练习的先决条件。

1. 要使用您的应用程序代码，您将其编译和打包成 JAR 文件。你可以用 SBT 编译和打包你的代码：

```
sbt assembly
```

2. 如果应用程序成功编译，则创建以下文件：

```
target/scala-3.2.0/sliding-window-scala-1.0.jar
```

上传 Apache Flink 直播 Scala 代码

在本节中，您将创建一个 Simple Storage S3 存储桶并上传您的应用程序代码。

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择创建存储桶
3. `ka-app-code-<username>`在存储段名称字段中输入。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。选择下一步。
4. 在配置选项中，保持设置不变，然后选择下一步。
5. 在设置权限中，保持设置不变，然后选择下一步。
6. 选择创建桶。
7. 选择`ka-app-code-<username>`存储桶然后选择 Upload（上载）。
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 `sliding-window-scala-1.0.jar` 文件。
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 桶中，您的应用程序可以在其中访问。

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 Scala test app**。
   - 对于 Runtime (运行时)，请选择 Apache Flink。
   - 将该版本保留为 Apache Flink 版本 1.15.2（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*

配置应用程序

可以按照以下步骤配置应用程序。

配置应用程序

1. 在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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，输入**sliding-window-scala-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 “属性” 下，选择 “添加群组”。
5. 输入以下信息:

组 ID	键	值
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

选择保存。

6. 在 “属性” 下，再次选择 “添加群组”。
7. 输入以下信息:

组 ID	键	值
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
10. 选择更新。

Note

当您选择启用亚马逊 CloudWatch 日志记录时，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`

编辑 IAM 策略

编辑 IAM 策略以添加权限以访问 Amazon S3 存储桶的权限。

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/sliding-window-scala-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"
        }
    ]
}
```

运行应用程序

可以通过运行应用程序、打开 Apache Flink 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

停止应用程序

要停止应用程序，请在MyApplication页面上选择停止。确认该操作。

创建并运行应用程序 (CLI)

在本节中，您将使用创建和运行 Kinesis Data Analytics 应用程序。Amazon Command Line Interface使用 kinesisanalyticsv2Amazon CLI 命令创建 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，您创建一个包含两个语句的权限策略：一个语句授予对源流进行读取操作的权限，另一个语句授予对接收流进行写入操作的权限。然后，您将策略附加到 IAM 角色（您在下一节中创建）。因此，当 Kinesis Data Analytics 担任该角色时，该服务具有从源流读取和写入接收流的必要权限。

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

```
{
    "ApplicationName": "sliding_window",
    "ApplicationDescription": "Scala sliding window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/KA-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "sliding-window-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-group:MyApplication:log-
stream:kinesis-analytics-log-stream"
      }
   ]
}
```

有关创建权限策略的 step-by-step 说明，请参阅 IAM 用户指南中的教程：创建并附加您的第一个客户托管策略。

创建 IAM 角色

在本节中，您将创建一个 IAM 角色，Kinesis Data Analytics 应用程序可以假定该角色来读取源流并写入接收流。

未经许可，Kinesis Data Analytics 无法访问您的直播。您可以通过 IAM 角色授予这些权限。每个 IAM 角色都附加了两个策略。信任策略授予 Kinesis Data Analytics 代入角色，权限策略决定了 Kinesis Data Analytics a

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，选择角色，然后选择创建角色。
3. 在 “选择可信身份类型” 下，选择 “Amazon服务”
4. 在 Choose the service that will use this role (选择将使用此角色的服务) 下，选择 Kinesis。
5. 在选择您的使用案例下，选择 Kinesis Analytics。
6. 选择Next: Permissions（下一步: 权限）。
7. 在 Attach permissions policies 页面上，选择 Next: Review。在创建角色后，您可以附加权限策略。
8. 在创建角色页面上**KA-stream-rw-role**，输入角色名称。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，更新角色的信任和权限策略
9. 将权限策略附加到角色。

   Note

   在本练习中，Kinesis Data Analytics 扮演这个角色，既可以从 Kinesis 数据流（源）读取数据，也可以将输出写入另一个 Kinesis 数据流。因此，您可以附加您在上一步中创建的策略，即创建权限策略。

   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。

有关创建角色的 step-by-step 说明，请参阅 IAM 用户指南中的创建 IAM 角色（控制台）。

创建应用程序

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（用户名）替换为您在上一节中选择的后缀。将服务执行角色中的示例账户 ID (012345678901) 替换为您的账户 ID。

```
{
    "ApplicationName": "sliding_window",
    "ApplicationDescription": "Scala sliding_window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/KA-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "sliding-window-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
```

```
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
                }
            },
            {
                "PropertyGroupId": "ProducerConfigProperties",
                "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
                }
            }
        ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
        "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-group:MyApplication:log-
stream:kinesis-analytics-log-stream"
      }
   ]
}
```

CreateApplication使用以下请求执行以创建应用程序：

```
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": "sliding_window",
    "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 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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": "sliding_window"
}
```

2. 使用上述请求执行StopApplication操作以停止应用程序：

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

应用程序现已停止。

添加 CloudWatch 日志选项

您可以使用Amazon CLI向您的应用程序添加 Amazon CloudWatch 日志流。有关在应用程序中使用 CloudWatch 日志的信息，请参阅设置应用程序日志。

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。在此示例中，您将更改源存储桶和目标直播的区域。

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": "sliding_window",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
      }
   }
}
```

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

当您需要使用新版本的代码包更新应用程序代码时，可以使用 UpdateApplicationCLI 操作。

Note

要加载具有相同文件名的新版本的应用程序代码，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。

要使用Amazon CLI，请从 Amazon S3 存储桶中删除您之前的代码包，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。<username>使用您在部分中选择的后缀更新存储创建相关资源 (p. 123)段名称后缀 ()。

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

清理 Amazon 资源

本节包括清理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 (p. 240)
- 删除 Kinesis Data Streams (p. 240)
- 删除您的Amazon S3 对象和存储桶 (p. 241)
- 删除您的 IAM 资源 (p. 241)
- 删除您的 CloudWatch 资源 (p. 241)

删除 Kinesis Data Analytics

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页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择操作，选择删除，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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

示例：在 Scala 中将流媒体数据发送到 Amazon S3

> **Note**
>
> 从 1.15 版本开始，Flink 是免费的 Scala。应用程序现在可以使用任何 Scala 版本的 Java API。Flink 仍然在一些关键组件中使用 Scala，但不会将 Scala 暴露给用户代码类加载器。因此，用户需要将 Scala 依赖项添加到他们的 jar 存档中。
> 有关 Flink 1.15 中 Scala 变更的更多信息，请参阅 One Feifteen 中的 Scala Free。

在本练习中，您将创建一个使用 Scala 3.2.0 和 Flink 的 Java DataStream API 的简单流媒体应用程序。该应用程序从 Kinesis 流读取数据，使用滑动窗口聚合数据，并将结果写入 S3。

> **Note**
>
> 要设置本练习所需的先决条件，请先完成入门 (Scala) 练习。您只需要在 Amazon S3 存储桶**data/**中创建一个额外的文件夹 ka-app-code-<username>.

本主题包含下列部分：

- 下载并检查应用程序代码 (p. 242)
- 编译并上传应用程序代码 (p. 242)
- 创建并运行应用程序（控制台）(p. 222)
- 创建并运行应用程序 (CLI) (p. 246)
- 更新应用程序代码 (p. 134)

- 清理 Amazon 资源 (p. 251)

下载并检查应用程序代码

此示例的 Python 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. 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/scala/S3Sink目录。

请注意有关应用程序代码的以下信息：

- build.sbt文件包含有关应用程序配置和依赖关系的信息，包括 Kinesis Data Analytics 库。
- 该BasicStreamingJob.scala文件包含定义应用程序功能的主要方法。
- 该应用程序使用 Kinesis 源从源码流中读取数据。以下片段创建了 Kinesis 源代码：

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

该应用程序还使用 a 写 StreamingFileSink 入 Amazon S3 存储桶的内容：

```
def createSink: StreamingFileSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val s3SinkPath =
 applicationProperties.get("ProducerConfigProperties").getProperty("s3.sink.path")

  StreamingFileSink
    .forRowFormat(new Path(s3SinkPath), new SimpleStringEncoder[String]("UTF-8"))
    .build()
}
```

- 应用程序创建源和接收连接器以使用 StreamExecutionEnvironment 对象访问外部资源。
- 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。读取运行时应用程序的属性以配置连接器。有关运行时属性的更多信息，请参阅运行时属性。

编译并上传应用程序代码

在本节中，您将编译应用程序代码并上传到 Amazon S3 存储桶。

编译应用程序代码

使用 SBT 编译工具为应用程序构建 Scala 代码。要安装 SBT，请参阅使用 cs 设置安装 sbt。您还需要安装 Java 开发工具包 (JDK)。请参阅完成练习的先决条件。

1. 要使用您的应用程序代码，您将其编译和打包成 JAR 文件。你可以用 SBT 编译和打包你的代码：

```
sbt assembly
```

2. 如果应用程序成功编译，则创建以下文件：

```
target/scala-3.2.0/s3-sink-scala-1.0.jar
```

上传 Apache Flink 直播 Scala 代码

在本节中，您将创建一个 Simple Storage S3 存储桶并上传您的应用程序代码。

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 选择创建存储桶
3. ka-app-code-<username>在存储段名称字段中输入。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。选择下一步。
4. 在配置选项中，保持设置不变，然后选择下一步。
5. 在设置权限中，保持设置不变，然后选择下一步。
6. 选择创建桶。
7. 选择ka-app-code-<username>存储桶然后选择 Upload（上载）。
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 s3-sink-scala-1.0.jar 文件。
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 桶中，您的应用程序可以在其中访问。

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。
   - 将该版本保留为 Apache Flink 版本 1.15.2（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*

配置应用程序

可以按照以下步骤配置应用程序。

配置应用程序

1. 在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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，输入**s3-sink-scala-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 “属性” 下，选择 “添加群组”。
5. 输入以下信息:

组 ID	键	值
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

   选择保存。
6. 在 “属性” 下，选择 “添加群组”。
7. 输入以下信息:

组 ID	键	值
ProducerConfigProperties	**s3.sink.path**	**s3a://ka-app-code-*<user-name>*/data**

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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
10. 选择更新。

> **Note**
>
> 当您选择启用亚马逊 CloudWatch 日志记录时，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`

编辑 IAM 策略

编辑 IAM 策略以添加权限以访问 Amazon S3 存储桶的权限。

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:Abort*",
                "s3:DeleteObject*",
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-<username>",
                "arn:aws:s3:::ka-app-code-<username>/*"
            ]
        },
        {
            "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"
```

```
        }
    ]
}
```

运行应用程序

可以通过运行应用程序、打开 Apache Flink 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

停止应用程序

要停止应用程序，请在MyApplication页面上选择停止。确认该操作。

创建并运行应用程序 (CLI)

在本节中，您将使用创建和运行 Kinesis Data Analytics 应用程序。Amazon Command Line Interface使用 kinesisanalyticsv2Amazon CLI 命令创建 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，您创建一个包含两个语句的权限策略：一个语句授予对源流进行读取操作的权限，另一个语句授予对接收流进行写入操作的权限。然后，您将策略附加到 IAM 角色（您在下一节中创建）。因此，当 Kinesis Data Analytics 担任该角色时，该服务具有从源流读取和写入接收流的必要权限。

使用以下代码创建 KAReadSourceStreamWriteSinkStream 权限策略。**username**替换为用于创建 Amazon S3 存储桶以存储应用程序代码的用户名。将亚马逊资源名称 (ARN) 中的账户 ID 替换**(012345678901)**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/getting-started-scala-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"
        }
    ]
}
```

有关创建权限策略的 step-by-step 说明，请参阅 IAM 用户指南中的教程：创建并附加您的第一个客户托管策略。

创建 IAM 角色

在本节中，您将创建一个 IAM 角色，Kinesis Data Analytics 应用程序可以假定该角色来读取源流并写入接收流。

未经许可，Kinesis Data Analytics 无法访问您的直播。您可以通过 IAM 角色授予这些权限。每个 IAM 角色都附加了两个策略。信任策略授予 Kinesis Data Analytics 代入角色，权限策略决定了 Kinesis Data Analytics a

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，选择角色，然后选择创建角色。
3. 在 “选择可信身份类型” 下，选择 “Amazon服务”
4. 在 Choose the service that will use this role (选择将使用此角色的服务) 下，选择 Kinesis。
5. 在选择您的使用案例下，选择 Kinesis Analytics。
6. 选择Next: Permissions（下一步: 权限）。
7. 在 Attach permissions policies 页面上，选择 Next: Review。在创建角色后，您可以附加权限策略。
8. 在创建角色页面上**KA-stream-rw-role**，输入角色名称。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，更新角色的信任和权限策略

9. 将权限策略附加到角色。

   Note

   在本练习中，Kinesis Data Analytics 扮演这个角色，既可以从 Kinesis 数据流（源）读取数据，也可以将输出写入另一个 Kinesis 数据流。因此，您可以附加您在上一步中创建的策略，即创建权限策略。

   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。

有关创建角色的 step-by-step 说明，请参阅 IAM 用户指南中的创建 IAM 角色（控制台）。

创建应用程序

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（用户名）替换为您在上一节中选择的后缀。将服务执行角色中的示例账户 ID (012345678901) 替换为您的账户 ID。

```
{
    "ApplicationName": "s3_sink",
    "ApplicationDescription": "Scala tumbling window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/KA-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "s3-sink-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "s3.sink.path" : "s3a://ka-app-code-<username>/data"
               }
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
```

```
   ]
}
```

CreateApplication使用以下请求执行以创建应用程序：

```
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": "s3_sink",
    "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 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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": "s3_sink"
}
```

2. 使用上述请求执行StopApplication操作以停止应用程序：

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

应用程序现已停止。

添加 CloudWatch 日志选项

您可以使用Amazon CLI向您的应用程序添加 Amazon CloudWatch 日志流。有关在应用程序中使用 CloudWatch 日志的信息，请参阅设置应用程序日志。

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。在此示例中，您将更改源存储桶和目标直播的区域。

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": "s3_sink",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "s3.sink.path" : "s3a://ka-app-code-<username>/data"
               }
            }
         ]
      }
   }
}
```

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

当您需要使用新版本的代码包更新应用程序代码时，可以使用 UpdateApplicationCLI 操作。

> **Note**
>
> 要加载具有相同文件名的新版本的应用程序代码，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。

要使用Amazon CLI，请从 Amazon S3 存储桶中删除您之前的代码包，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。<username>使用您在部分中选择的后缀更新存储创建相关资源 (p. 123)段名称后缀 ()。

```
{
    "ApplicationName": "s3_sink",
    "CurrentApplicationVersionId": 1,
```

```
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "s3-sink-scala-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

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

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页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择操作，选择删除，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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

Amazon Kinesis Data Analytics

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。责任共担模式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，请参阅合规性计划范围内的Amazon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 服务来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 for Apach Flink

静态加密

请注意以下有关使用 Apache Flink Data Analytics 对静态数据进行加密的内容：

- 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？。
- 可以使用 Kinesis Data Firehose 对输出数据进行静态加密，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 托管密钥的服务器端加密 (SSE-KMS) 保护数据。
- 适用于 Apache FlAmazon Kinesis Data Analytics 可以从任何流媒体源读取，也可以写入任何流媒体或数据库目的地。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。

- 持久的应用程序存储是静态加密的。
- 正在运行的应用程序存储是静态加密的。

传输中加密

Kinesis Data Analytics 对传输中的所有数据进行加密。所有 Kinesis Data Analytics 应用程序均启用传输中加密，无法禁用。

Kinesis Data Analytics 会在以下情况下加密传输中的数据：

- 从 Kinesis Data Analytics 传输到 Kinesis Data Analyt
- 在 Kinesis Data Analytics 内部组件之间传输的数据。
- 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管理密钥。客户管理的密钥不受支持。

Amazon Kinesis Data Analytics

Amazon Identity and Access Management (IAM) 是一项 Amazon Web Service，可以帮助管理员安全地控制对 Amazon 资源的访问。IAAAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）来使用 Kinesis Data Analytics IAM 是一项无需额外费用即可使用的 Amazon Web Service。

主题

受众

使用Amazon Identity and Access Management (IAM) 的方式因您可以在 Kinesis Data Analytics 中执行的操作而异。

服务用户-如果您使用 Kinesis Data Analytics e 来完成工作，则您的管理员会为您提供所需的凭证和权限。当您使用更多 Kinesis Data Analytics 功能来完成工作时，您可能需要更多权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 Kinesis Data Analytics 中的一项功能，请参阅对 Amazon Kinesis Data Analytics (p. 263)。

服务管理员 — 如果您在公司负责管理 Kinesis Data Analytics 资源，您可能对 Kinesis Data Analytics 您有责任确定您的服务用户应访问哪些 Kinesis Data Analytics 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Kinesis Data Analytics 搭配使用的更多信息，请参阅Amazon Kinesis Data Analytics 如何使用 (p. 257)。

IAAAM 管理员-如果您是 IAAAM 管理员，您可能希望了解如何编写策略以管理对 Kinesis Data Analytics 要查看您可在 IAM 中使用的 Kinesis Data Analytics 的示例策略，请参阅适用于 Amazon Kinesis Data Analytics 的基于身份 (p. 262)。

使用身份进行身份验证

身份验证是您使用身份凭证登录 Amazon 的方法。您必须作为 Amazon Web Services 账户根用户、IAM 用户或通过代入 IAM 角色进行身份验证（登录到 Amazon）。

如果您以编程方式访问 Amazon，则 Amazon 将提供软件开发工具包 (SDK) 和命令行界面（CLI），以便使用您的凭证对您的请求进行加密签名。如果您不使用 Amazon 工具，则必须自行对请求签名。有关使用推荐的方法自行对请求签名的更多信息，请参阅 Amazon 一般参考中的签名版本 4 签名流程。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如，Amazon 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《IAM 用户指南》中的在 Amazon 中使用多重身份验证 (MFA)。

Amazon Web Services 账户根用户

当您创建 Amazon Web Services 账户时，最初使用的是一个对账户中所有 Amazon Web Services和资源拥有完全访问权限的登录身份。此身份称为 Amazon Web Services 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅 Amazon Account Management 参考指南 中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）结合使用联合身份验证和身份提供程序，以使用临时凭证来访问 Amazon Web Services。

联合身份是来自企业用户目录、Web 身份提供程序、Amazon Directory Service、的用户，或任何使用通过身份源提供的凭证来访问 Amazon Web Services 的用户。当联合身份访问 Amazon Web Services 账户时，他们代入角色，而角色提供临时凭证。

IAM 用户和组

IAM 用户是Amazon Web Services 账户内对某个人员或应用程序具有特定权限的一个身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是Amazon Web Services 账户中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 Amazon Web Services Management Console 中暂时代入 IAM 角色。您可以调用 Amazon CLI 或 Amazon API 操作或使用自定义 URL 以担任角色。有关使用角色的方法的更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- Federated user access（联合用户访问）– 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的为第三方身份提供商创建角色。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户访问 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 Amazon Web Services，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 – 某些 Amazon Web Services使用其它 Amazon Web Services中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Simple Storage Service（Amazon S3）中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 主体权限 – 当您使用 IAM 用户或角色在 Amazon 中执行操作时，您将被视为主体。策略向主体授予权限。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中触发另一个操作。在这种情况下，您必须具有执行这两个操作的权限。要查看某个操作是否在策略中需要其他相关操作，请参阅《服务授权参考》中的 Amazon Kinesis Data Analytics 的操作、资源和条件键。
  - 服务角色 – 服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅 IAM 用户指南中的创建向 Amazon Web Service 委派权限的角色。
  - 服务相关角色 – 服务相关角色是与 Amazon Web Service关联的一种服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 Amazon Web Services 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 Amazon CLI 或 Amazon API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 Amazon 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM 用户指南中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅 IAM 用户指南中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 Amazon 身份或资源，以控制 Amazon 中的访问。策略是 Amazon 中的对象；在与身份或资源相关联时，策略定义它们的权限。在主体（用户、根用户或角色会话）发出请求时，Amazon 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 Amazon 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南 中的 JSON 策略概览。

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。然后，管理员可以向角色添加 IAM 策略，并且用户可以代入角色。

IAM policy 定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的 创建 IAM policy。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管式策略是可以附加到Amazon Web Services 账户中的多个用户、组和角色的独立策略。托管式策略包括 Amazon 托管式策略和客户托管式策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅 IAM 用户指南 中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合身份用户或 Amazon Web Services。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用来自 IAM 的 Amazon 托管式策略。

访问控制列表 (ACL)

访问控制列表 (ACL) 控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Simple Storage Service（Amazon S3）、Amazon WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的访问控制列表 (ACL) 概览。

其它策略类型

Amazon 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 – 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅 IAM 用户指南中的 IAM 实体的权限边界。
- 服务控制策略 (SCP) – SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 Amazon Organizations 中的最大权限。Amazon Organizations 服务可以分组和集中管理您的企业拥有的多个 Amazon Web Services 账户 账户。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体（包括每个 Amazon Web Services 账户根用户）的权限。有关 Organizations 和 SCP 的更多信息，请参阅 Amazon Organizations 用户指南中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解 Amazon 如何确定在涉及多种策略类型时是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amazon Kinesis Data Analytics 如何使用

在使用 IAM 管理对 Kinesis Data Analytics 的访问之前，了解哪些 IAM 功能可与 Kinesis Data Analytics 协同工作。

AM Kinesis Data Analytics 可以使用 IAM

IAM 功能	Kinesis Data Analytics
基于身份的策略 (p. 258)	是
基于资源的策略 (p. 258)	否
策略操作 (p. 259)	是
策略资源 (p. 259)	是
策略条件键 (p. 260)	否
ACL (p. 260)	否
ABAC（策略中的标签） (p. 260)	是
临时凭证 (p. 261)	是
委托人权限 (p. 261)	是
服务角色 (p. 261)	否
服务相关角色 (p. 262)	否

要大致了解 Kinesis Sis Data Analytictics 和其他Amazon服务如何与大多数 IAAAM 功能配合使用，请参阅 IAAAM 用户指南中的与 IAAAM 一起使用的Amazon服务。

Kinesis Data Analytics 的基于身份的策略

支持基于身份的策略	是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM policy。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

Kinesis Data Analytics 的基于身份的策略示例

要查看 Kinesis Data Analytics 的基于身份的策略的示例，请参阅适用于 Amazon Kinesis Data Analytics 的基于身份 (p. 262)。

Kinesis Data Analytics 内基于资源

支持基于资源的策略。	是

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在

什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合身份用户或 Amazon Web Services。

要启用跨账户存取，您可以将整个账户或其它账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当主体和资源处于不同的Amazon Web Services 账户中时，则信任账户中的 IAM 管理员还必须授予主体实体（用户或角色）对资源的访问权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。

Kinesis Data Analytics

支持策略操作	是

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么 条件 下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 Amazon API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行相关操作的权限。

要查看 Kinesis Data Analytics 操作列表，请参阅服务授权参考中的 Amazon Kinesis Data Analytics 定义的操作。

Kinesis Data Analytics 在操作前使用以下前缀：

```
Kinesis Analytics
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "Kinesis Analytics:action1",
      "Kinesis Analytics:action2"
         ]
```

您也可以使用通配符（*）指定多个操作。例如，要指定以单词 `Describe` 开头的所有操作，包括以下操作：

```
"Action": "Kinesis Analytics:Describe*"
```

要查看 Kinesis Data Analytics 的基于身份的策略的示例，请参阅适用于 Amazon Kinesis Data Analytics 的基于身份 (p. 262)。

Kinesis Data Analytics

支持策略资源	是

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符（*）指示语句应用于所有资源。

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

要查看 Kinesis Data Analytics 资源类型及其 ARN 的列表，请参阅服务授权参考中的 Amazon Kinesis Data Analytics 定义的资源。要了解您可以使用哪些操作指定每个资源的 ARN，请参阅 Amazon Kinesis Data Analytics 定义的操作。

要查看 Kinesis Data Analytics 的基于身份的策略的示例，请参阅适用于 Amazon Kinesis Data Analytics 的基于身份 (p. 262)。

Kinesis Data Analytics 的策略条件密钥

支持特定于服务的策略条件键	是

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 Amazon 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则 Amazon 使用逻辑 OR 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM policy 元素：变量和标签。

Amazon支持全局条件键和特定于服务的条件键。要查看所有 Amazon 全局条件键，请参阅《IAM 用户指南》中的 Amazon 全局条件上下文键。

要查看 Kinesis Data Analytics 条件密钥列表，请参阅服务授权参考中的 Amazon Kinesis Data Analytics 条件密钥。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon Kinesis Data Analytics 定义的操作。

要查看 Kinesis Data Analytics 的基于身份的策略的示例，请参阅适用于 Amazon Kinesis Data Analytics 的基于身份 (p. 262)。

Kinesis Data Analytics 中的访问控制列表（ACL）

支持 ACL	否

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

使用 Kinesis Data Analytics 的基于属性的访问控制 (ABAC)

支持 ABAC（策略中的标签）	是

基于属性的访问控制（ABAC）是一种授权策略，该策略基于属性来定义权限。在 Amazon 中，这些属性称为标签。您可以将标签附加到 IAM 实体（用户或角色）以及Amazon资源。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为 Yes（是）。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为 Partial（部分）。

有关 ABAC 的更多信息,请参阅《IAM 用户指南》中的什么是 ABAC？。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问控制（ABAC）。

使用 Kinesis Data Analytics

支持临时凭证	是

某些 Amazon Web Services在您使用临时凭证登录时无法正常工作。有关更多信息，包括 Amazon Web Services与临时凭证配合使用，请参阅 IAM 用户指南中的使用 IAM 的 Amazon Web Services。

如果您不使用用户名和密码而用其它方法登录到Amazon Web Services Management Console，则使用临时凭证。例如，当您使用贵公司的单点登录（SSO）链接访问Amazon时，该过程将自动创建临时凭证。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的切换到角色（控制台）。

您可以使用 Amazon CLI 或者 Amazon API 创建临时凭证。之后，您可以使用这些临时凭证访问 Amazon。Amazon建议您动态生成临时凭证，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

Kinesis Data Analytics 跨服务主体权限

支持委托人权限	是

当您使用 IAM 用户或角色在Amazon中执行操作时，您将被视为委托人。策略向主体授予权限。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中触发另一个操作。在这种情况下，您必须具有执行这两个操作的权限。要查看某个操作是否在策略中需要其他相关操作，请参阅《服务授权参考》中的 Amazon Kinesis Data Analytics 的操作、资源和条件键。

Kinesis Data Analytics

支持服务角色	是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅 IAM 用户指南中的创建向 Amazon Web Service 委派权限的角色。

> Warning
>
> 更改服务角色的权限可能会破坏 Kinesis Data Analytics 的功能。仅当 Kinesis Data Analytics 提供相关指导时才编辑服务角色。

Kinesis Data Analyts 的服务相关角色

支持服务相关角色	是

服务相关角色是一种与 Amazon Web Service相关的服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 Amazon Web Services 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅能够与 IAM 搭配使用的Amazon服务。在表中查找服务相关角色列中包含 Yes 额表。选择 Yes 链接以查看该服务的服务相关角色文档。

适用于 Amazon Kinesis Data Analytics 的基于身份

默认情况下，用户和角色没有创建或修改 Kinesis Data Analytics 资源的权限。他们也无法使用 Amazon Web Services Management Console、Amazon Command Line Interface（Amazon CLI）或 Amazon API 执行任务。要用户用户在其需要的资源上执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的 创建 IAM policy。

有关 Kinesis Data Analytics 定义的操作和资源类型的详细信息，包括每种资源类型的 ARN 格式，请参阅服务授权参考中的 A mazon Kinesis Kinesis Data Analytictictices Data Analytictictices Data Analytictictic es

主题

- 策略最佳实践 (p. 262)
- 使用 Kinesis Data Analytics (p. 263)
- 允许用户查看他们自己的权限 (p. 263)

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 Kinesis Data Analytics 资源。这些操作可能会使 Amazon Web Services 账户 产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- Amazon 托管策略及转向最低权限许可入门 - 要开始向用户和工作负载授予权限，请使用 Amazon 托管策略来为许多常见使用场景授予权限。您可以在 Amazon Web Services 账户 中找到这些策略。我们建议通过定义特定于您的使用场景的 Amazon 客户管理型策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 Amazon 托管策略或工作职能的 Amazon 托管策略。
- 应用最低权限 – 在使用 IAM policy 设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM policy 中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果通过特定 Amazon Web Service（例如 Amazon CloudFormation）使用服务操作，您还可以使用条件来授予对服务操作的访问权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM policy语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- Require multi-factor authentication（MFA）[需要多重身份验证（MFA）] – 如果您所处的场景要求您的 Amazon Web Services 账户 中有 IAM 用户或根用户，请启用 MFA 来提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实践的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 Kinesis Data Analytics

要访问 Amazon Kinesis Data Analytics 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您中 Kinesis Data Analytics 资源的详细信息Amazon Web Services 账户。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于只需要调用 Amazon CLI 或 Amazon API 的用户，无需为其提供最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍可使用 Kinesis Data Analytics es 控制台，还请将 Kinesis Data AnalyticsConsoleAccess sis Data AnalyticsReadOnlyAmazon 有关更多信息，请参阅 IAM 用户指南中的为用户添加权限。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上完成此操作或者以编程方式使用 Amazon CLI 或 Amazon API 所需的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

对 Amazon Kinesis Data Analytics

使用以下信息可帮助您诊断和修复在使用 Kinesis Data Analytics 和 IAM 时可能遇到的常见问题。

主题

我无权在 Kinesis Data Analytics 中执行操作

如果 Amazon Web Services Management Console 告诉您，您无权执行某个操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。

当 `mateojackson` 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `Kinesis Analytics:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: Kinesis
 Analytics:GetWidget on resource: my-example-widget
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `Kinesis Analytics:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

我无权执行 im：PassRole

如果您收到一个错误，表明您无权执行操作，则必须更新策略以允许您将角色传递给 Kinesis Sis Sis Data Analytictictics Data Analytictictic

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. 268)：

Amazon Kinesis Data Analytics 的合规性验证

作为多项合规性计划的一部分，第三方审计员将评估 Amazon Kinesis Data Analytics for Apache Flink 的安全Amazon性和合规性。其中包括 SOC、PCI、HIPAA 等。

有关特定合规性计划范围内的Amazon服务列表，请参阅合规性计划范围内的 Amazon Web Services Service。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 Analytictics for Apache Flink Data Analytictics fornick Data Analytictictics for Apache Flink 如果您对 Kinesis Data Analytics for Apache Flink 的使用需遵守 HIPAA 或 PCI 等标准，将Amazon提供以下有用资源：

- 《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

AmazonFedRAMP 合规性计划包括作为 FedRAMP 授权服务的 Kinesis Data Analytics。如果您是联邦或商业客户，您可以使用该服务在（美国）区域以及具有高影响力级别数据的Amazon GovCloud （美国）区域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（弗吉尼亚北部）、美国东部（弗吉尼亚北部）、美国东部（弗吉尼亚北部）、美国东部（弗吉尼亚北部）、美国东部（弗吉尼亚北部）、数据达到中等级别。

您可以通过Amazon FedRAMP PMO 或您的Amazon销售客户经理请求访问 FedRAMP 安全包，也可以通过 Artifact Artifact 上的 ArAmazon tifactAmazon 下载这些安全包。

有关更多信息，请参阅 FedRAMP。

Amazon Kinesis Data Analytics for Apach

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Amazon全球基础设施之外，Apache Flink Kinesis Data Analytictics Flink 还提供多种功能，以帮助支持您的数据恢复能力和备份需求。

灾难恢复

Kinesis Data Analytics 在无服务器模式下运行，并通过执行自动迁移来处理主机降级、可用区可用性和其他与基础设施相关的问题。Kinesis Data Analytics 通过多种冗余机制实现了这一目标。每个使用 Apache Flink 的 Kinesis Data Analytics 应用程序都在单租户 Apache Flink 集群中运行。Apache Flink 集群 JobMananger 在高可用性模式下使用 Zookeeper 在多个可用区内运行。Kinesis Data Analytics 使用亚马逊 EKS 部署 Apache Flink。亚马逊 EKS 中跨可用区的每个Amazon区域使用多个 Kubernetes Pod。如果出现故障，Kinesis Data Analytics 首先尝试使用应用程序的检查点（如果有）在正在运行的 Apache Flink 集群中恢复应用程序。

适用于 Apache 的 Kinesis Data Analytics Flink 使用检查点和快照备份应用程序状态：

- 检查点是 Kinesis Data Analytics 定期自动创建的应用程序状态的备份，用于从故障中恢复。
- 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. 27)。

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。
- 使用CreateApplicationSnapshot操作SnapshotName参数对@@ 保存点进行版本控制。

Kinesis Data Analytics 对存储在检查点和保存点中的数据进行加密。

AmazKinesis Data Analytics 中的基础架构安全

作为一项托管式服务，Amazon Kinesis Data Analytics a AnAmazon alytictices Data Analytictictics Data Analytictictices

您可以使用Amazon发布的 API 调用通过网络访问 Kinesis Data Analytics。对 Kinesis Data Analytics 的所有 API 调用均通过传输层安全 (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。

Amazis 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凭证才能访问其他服务中的资源，例如 Kinesis 数据流、Kinesis Data Firehose 交付流或 Amazon S3 存储桶。您不应直接在应用程序或 Amazon S3 桶中存储 Amazon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 角色来管理您的应用程序访问其他资源的临时凭证。在使用角色时，您不需要使用长期凭证来访问其他资源。

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。您应该在依赖资源中实现服务器端加密，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. 253)。

CloudTrail 用于监控 API 调用

Kinesis Data Analytictics 与Amazon CloudTrail集成，后者是一项提供 Kinesis Service、角色或 Amazon 服务在 Kinesis istis Data Analytictices Data

通过使用收集的信息 CloudTrail，您可以确定向 Kinesis Data Analytics（分析）、发出请求的 IP 地址、请求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. 298)：

Amazon Kinesis Data Analytics 中的记录和监控

监控是保持 Amazon Kinesis Sis Kinesis Data Analytictictics 和 Kinesis Sis Sis Data Analytictices 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，您应该创建一个监控计划，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，在您的环境中建立正常 Kinesis Data Analytics 性能的基准。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。在监控 Kinesis Data Analytics 时，您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故障非常重要。但是，日志子系统需要收集日志条目并将其转发到 Lo CloudWatch gs 虽然有些日志记录很好而且很理想，但大量的日志记录会使服务超负荷并导致 Flink 应用程序落后。记录异常和警告无疑是个好主意。但是你无法为 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处理过的消息生成日志输出，请在 Flink 应用程序 DataStream 内部使用一个额外的接收器和一个适当的接收器将数据发送到 Amazon S3 或 CloudWatch. 请勿将 Java 日志系统用于此目的。此外，Kinesis Data Analytics的`Debug Monitoring Log Level`设置会产生大量流量，这可能会产生背压。您只能在积极调查应用程序问题时使用它。

使用日志见解查询 CloudWatch 日志

CloudWatch Logs Insights 是一项用于大规模查询日志的强大服务。客户应利用其功能快速搜索日志，以识别和减少操作事件期间的错误。

以下查询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
```

有关其他有用的查询，请参阅示例查询。

监控

在生产环境中运行流媒体应用程序时，您打算持续无限期地执行该应用程序。实现对所有组件的监控和适当警报至关重要，而不仅仅是 Flink 应用程序。否则，你就有可能在早期错过新出现的问题，只有在操作事件完全解开且难以缓解时才意识到这一点。需要监控的一般内容包括：

- 来源在摄取数据吗？
- 数据是从源读取的（从源的角度来看）吗？
- Flink 应用程序在接收数据吗？
- Flink 应用程序能够跟上步伐还是落后了？
- Flink 应用程序是否将数据保存到接收器中（从应用程序的角度来看）？
- 接收器在接收数据吗？

然后应该为 Flink 应用程序考虑更具体的指标。此CloudWatch 仪表板提供了一个很好的起点。有关生产应用程序要监控哪些指标的更多信息，请参阅在 Amazon Kinesis Data Analytics 中使用 CloudWatch 警报 (p. 290)。这些指标包括：

- records_lag_max 和 millsBehindLatest— 如果应用程序使用来自 Kinesis 或 Kafka，则这些指标表明应用程序是否落后并需要进行扩展才能跟上当前负载。这是一个很好的通用指标，对于所有类型的应用程序都易于跟踪。但它只能用于反应式扩展，即当应用程序已经落后时。
- cpuUt ilization 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 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检查点需要多长时间。如果检查点增加或持续很长时间，则应用程序会持续在检查点上花费时间，并且实际处理的周期会减少。在某些时候，检查点可能会扩大，或者需要很长时间才能失效。除了监控绝对值外，客户还应考虑使用`RATE(lastCheckpointSize)`和监控变化率`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量。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`RATE(numberOfFailedCheckpoints)`通过监控来警报梯度而不是绝对值。

设置应用程序日志记录

通过向您的 Kinesis Data Analytics 应用程序添加 Amazon CloudWatch 日志选项，您可以监控应用程序事件或配置问题。

本主题介绍如何将应用程序配置为将应用程序事件写入 CloudWatch 日志流。 CloudWatch 日志选项是应用程序设置和权限的集合，您的应用程序使用这些设置和权限来配置将应用程序事件写入 CloudWatch 日志的方式。您可以使用Amazon Web Services Management Console或Amazon Command Line Interface (Amazon CLI) 添加和配置 CloudWatch 日志记录选项。

请注意以下有关向应用程序添加 CloudWatch 日志记录选项的注意事项：

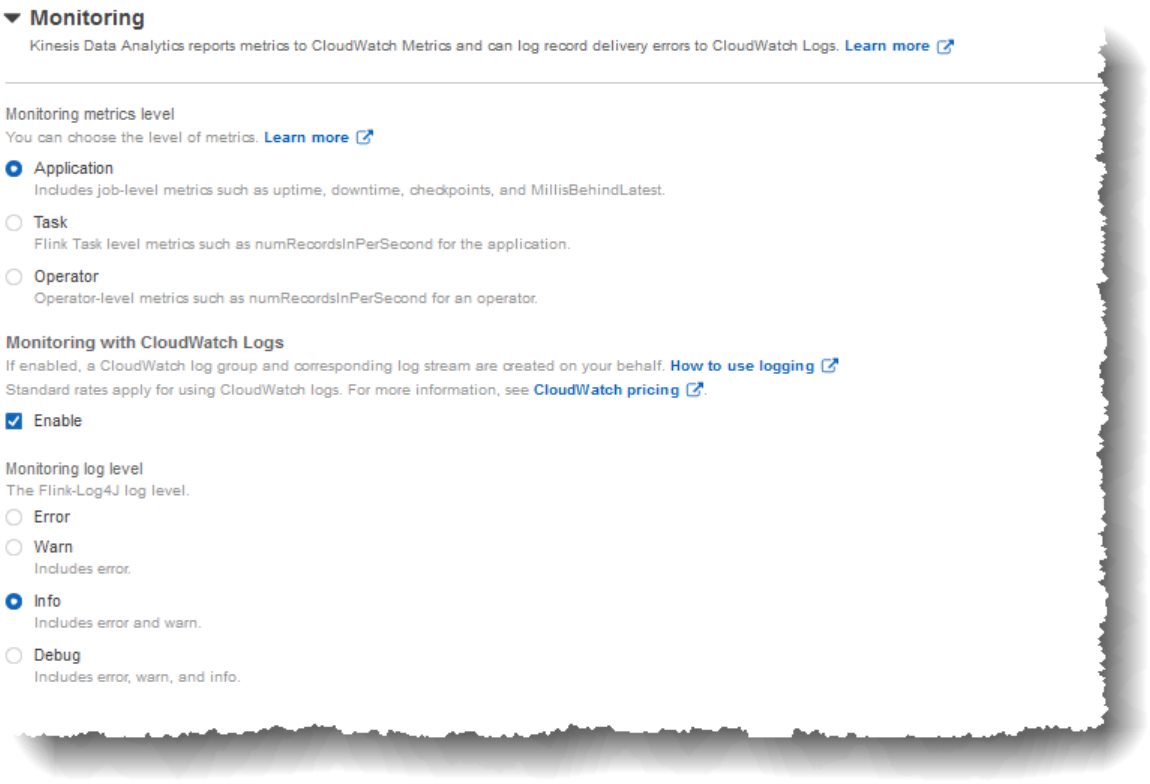
- 当您使用控制台添加 CloudWatch 日志记录选项时，Kinesis Data Analytics 会为您创建 CloudWatch 日志组和日志流，并将您的应用程序写入日志流所需的权限。
- 使用 API 添加 CloudWatch 日志选项时，还必须创建应用程序的日志组和日志流，并添加应用程序写入日志流所需的权限。

本主题包含下列部分：

使用控制台设置 CloudWatch 日志

当您在控制台中为应用程序启用 CloudWatch CloudWatch 日志记录时，将为您创建日志组和日志流。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 “配置应用程序” 页面中的CloudWatch 日志设置。



Kinesis Data Analytics 使用以下约定创建一个命名的日志组，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日志流。

```
kinesis-analytics-log-stream
```

您可以使用配置应用程序页面的监控日志级别部分设置应用程序监控指标级别和监控日志级别。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. 274)。

使用 CLI 设置 CloudWatch 日志

要使用添加 CloudWatch 日志选项Amazon CLI，请执行以下操作：

- 创建 CloudWatch 日志组和日志组。
- 使用操作创建应用程序时添加日志记录选项，或使用CreateApplication操作向现有应用程序添加日志选项。AddApplicationCloudWatchLoggingOption
- 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

您可以创建 CloudWatch 日志组并使用 CloudWatch 日志控制台或 API 进行流式传输。有关创建 CloudWatch 日志组和日志流的信息，请参阅使用日志组和日志流。

使用应用程序 CloudWatch 日志选项

使用以下 API 操作向新应用程序或现有应用程序添加日志选项，或更改现有应用程序的日志选项。CloudWatch 有关如何将 JSON 文件用于 API 操作输入的信息，请参阅 Kinesis Data Analytics (p. 423)。

在创建应用程序时添加 CloudWatch 日志选项

下面的示例演示如何在创建应用程序时使用CreateApplication操作添加 CloudWatch 日志选项。在示例中，使用您自己的信息替换*######### CloudWatch ##### Amazon #### (ARN)*。有关该操作的更多信息，请参阅 CreateApplication。

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "test-application-description",
    "RuntimeEnvironment": "FLINK-1_15",
    "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>"
 }]
}
```

向现有应用程序添加 CloudWatch 日志选项

下面的示例演示如何使用AddApplicationCloudWatchLoggingOption操作将 CloudWatch 日志选项添加至现有应用程序。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 CloudWatch 日志选项

下面的示例演示如何使用UpdateApplication操作修改现有 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>
}
```

从应用程序中删除 CloudWatch 日志选项

下面的示例演示如何使用DeleteApplicationCloudWatchLoggingOption操作删除现有 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级别，请使用CreateApplication操作的MonitoringConfigurationMonitoringConfigurationUpdate参数或UpdateApplication操作的参数。

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. 274)。

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_15",
   "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"
         }
      }
   }
}
```

添加写入 CloudWatch 日志流的权限

Kinesis Data Analytics 需要权限才能将错误配置错误写入 CloudWatch。您可以将这些权限添加到 Kinesis Data Analytics 担任的Amazon Identity and Access Management (IAM) 角色。

有关使用 IAM 角色进行Kinesis Data Analytics 的更多信息，请参阅Amazon Kinesis Data Analytics (p. 254)。

信任策略

要授予 Kinesis Data Analytics 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

要向应用程序授予 CloudWatch 从 Kinesis Data Analytics 资源写入日志事件的权限，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. 33)。
- 操作符：指标范围是每个操作符。有关操作符的信息，请参阅the section called “DataStream API 操作员” (p. 18)。
- 并行度：指标范围是应用程序并行度。您只能使用 UpdateApplicationAPI 的 MonitoringConfigurationUpdate参数设置此指标级别。您无法使用控制台设置此指标级别。有关并行度的信息，请参阅the section called “扩缩” (p. 33)。

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。
- 信息：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。
- 调试：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：

- 验证您的应用程序的 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 “添加写入 CloudWatch 日志流的权限” (p. 273)：
- 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，或使用DescribeApplication或ListApplications操作。
- 监控 CloudWatch 指标downtime，例如诊断其他应用程序问题。有关读取 CloudWatch 指标的信息，请参阅指标与维度 (p. 278)。

下一个步骤

在应用程序中启用 CloudWatch 登录功能后，您可以使用 CloudWatch Logs Insights 来分析应用程序日志。有关更多信息，请参阅the section called “分析日志” (p. 275)：

使用日志见解分析 CloudWatch 日志

按照上一节所述向应用程序添加 CloudWatch 日志记录选项后，您可以使用 Lo CloudWatch gs Insights 查询日志流中是否存在特定事件或错误。

CloudWatch 您可以使用 Logs Insights，通过交互方式搜索并分析 Log CloudWatch s 中的日志数据。

有关开始使用 Lo CloudWatch gs Insights 的信息，请参阅使用 Logs Insights 分析 CloudWatch 日志数据。

运行示例查询

本节介绍如何运行示例 CloudWatch Logs Insights 查询。

先决条件

- 在 Logs 中设置的现有日志组和 CloudWatch 日志流。
- 存储在 Logs 中的现有 CloudWatch 日志。

如果您使用服务（如Amazon CloudTrail、Amazon Route 53 或 Amazon VPC），则您可能已将日志设置为从这些服务转到 Logs，您可能已将日志设置为从这些服务转到 CloudWatch Logs，有关将事件发送到 Lo CloudWatch gs 的更多信息，请参阅 CloudWatch 日志入门。

Lo CloudWatch gs Insights Insights Insights 中的查询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运算的结果。本节说明了一个返回一组日志事件的查询。

运行 CloudWatch Logs Insights 示例查询

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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。

   当您选择日志组时，Logs Insights Insights Insights Insights Insightsights s（发现的字段），并将其显示在右侧窗格 CloudWatch 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 CloudWatch Logs Insights 查询的更多信息，请参阅运行和修改示例查询。

示例查询

本部分包含 CloudWatch 用于分析 Kinesis Data Analytics 应用程序日志的 Logs Insights 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
>
> 将以下查询示例中的区域 (*us-west-2*)、账户 ID (*012345678901 YourApplication*) 和应用程序名称 () 替换为应用程序的区域和账户 ID。

本主题包含下列部分：

- 分析操作：分配任务 (p. 276)
- 分析操作：更改并行度 (p. 277)
- 分析错误：访问被拒绝 (p. 277)
- 分析错误：找不到源或接收器 (p. 277)
- 分析错误：应用程序的任务相关故障 (p. 277)

分析操作：分配任务

以下 L CloudWatch ogs Insights 查询返回 Apache Flink Job 管理器在任务管理器之间分配的任务数量。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. 33)。

```
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 Logs Insights 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度的更多信息，请参阅 Apache Flink 文档中的设置并行度：运算符级别。

```
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 Apache Flink 文档中的作业和调度。

分析操作：更改并行度

以下 CloudWatch Logs Insights 查询返回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. 35)。

```
fields @timestamp, @parallelism
| filter message like /property: parallelism.default, /
| parse message "default, *" as @parallelism
| sort @timestamp asc
```

分析错误：访问被拒绝

以下 CloudWatch Logs Insights 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 Logs Insights 查询返回`ResourceNotFound`日志。 `ResourceNotFound`如果找不到 Kinesis 源或接收器，则会记录结果。

```
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 L CloudWatch ogs Insights 查询返回应用程序与任务相关的故障日志。如果应用程序的状态从`RUNNING`切换为，则会生成这些日志`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 及更早版本的应用程序，与任务相关的故障将导致应用程序状态从`FAILED`改`RUNNING`为切换。使用 Apache Flink 1.8.2 及更早版本时，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：

当你的 Amazon Kinesis Data Analytics for Apache Flink 处理数据源时，Kinesis Data Analytics 会向亚马逊报告以下指标和维度 CloudWatch。

应用程序指标

指标	Unit	描述	Level	使用说明	
backPressuredTimeMsPerSecond*	毫秒	每秒向该任务或操作员施加反压的时间（以毫秒为单位）。	任务、操作员、并行性	*仅适用于运行 Flink 版本 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（既非空闲也非后压）的时间（以毫秒为单位）。如果无法计算该值，则可以为 NaN。	任务、操作员、并行性	*仅适用于运行 Flink 版本 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	任务管理器中 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该指标的五个样本。	应用程序	您可以使用此指标来监控应用程序中的最小、平均和最大 CPU 利用率。该CPUUtilization指标仅考虑容器内运行的 TaskManager JVM 进程的 CPU 使用率。	
containerCPUUtilization	百分比	Flink 应用程序集群中各任务管理器容器的 CPU 利用率的	应用程序	每个集装箱的计算公式为：	

指标	Unit	描述	Level	使用说明	
		总百分比。例如，如果有五个任务管理器，则相应地有五个 TaskManager 容器，Kinesis Data Analytics 每 1 分钟报告间隔发布 2* 5 个该指标的样本。		容器消耗的 CPU 总时间（以秒为单位）* 100 /容器 CPU 限制（以 CPU/秒为单位） 该CPUUtilization指标仅考虑容器内运行的 TaskManager JVM 进程的 CPU 使用率。在同一容器中，还有其他组件在 JVM 之外运行。该containerCPUUtilization指标可以让你更全面地了解容器的 CPU 消耗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内存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，Kinesis Data Analytics 每 1 分钟报告间隔发布 2* 5 个该指标的样本。	应用程序	每个集装箱的计算公式为： 容器内存使用量（字节）* 100 / 根据 pod 部署规范，容器内存限制（以字节为单位） HeapMemoryUtilization和ManagedM标仅考虑特定的内存指标，例如 TaskManager JVM 的堆内存使用率或托管内存（RocksDB State Backend 等原生进程在 JVM 之外的内存使用情况）。该containerMemoryUtilization指标包括工作集内存，可以更好地跟踪总内存消耗情况，从而可以更好地跟踪总内存消耗情况。当它耗尽时，它会Out of Memory Error进入 TaskManager 吊舱。	
containerDiskUtilization	百分比	Flink 应用程序集群中各任务管理器容器的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，Kinesis Data Analytics 每 1 分钟报告间隔发布 2* 5 个该指标的样本。	应用程序	每个集装箱的计算公式为： 以字节为单位的磁盘使用量 * 100 /容器的磁盘限制（以字节为单位） 对于容器，它表示在其中设置容器根卷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、运算符、任务、并行性	此记录仅针对具有两个输入的维度发出。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、运算符、任务、并行性		
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启的总次数。该指标不衡量细粒度的重启。	应用程序	您可以使用此指标来评估应用程序的总体运行状况。在 Kinesis Data Analytics 的内部维护期间，可能会重新启动。重启频率高于正常值可能表示应用程序出现问题。	
heapMemoryUtilization	百分比	各任务管理器的总体堆内存利用率。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该指标的五个样本。	应用程序	您可以使用此指标来监控应用程序中的最小、平均和最大堆内存利用率。HeapMemoryUtilization仅考虑特定的内存指标，例如 TaskManager JVM 的堆内存使用情况。	
idleTimeMsPerSecond*	毫秒	此任务或操作员每秒处于空闲状态（没有要处理的数据）的时间（以毫秒为单位）。空闲时间不包括反向加压时间，因此，如果任务受到反向压力，则该任务不会处于空闲状态。	任务、操作员、并行性	*仅适用于运行 Flink 版本 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、运算符、任务、并行性	*仅适用于运行 Flink 版本 1.13 的 KDA 应用程序。 这与 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、运算符、任务、并行性	*仅适用于运行 Flink 版本 1.13 的 KDA 应用程序。 这与 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。该ManagedMemoryUtilzations指标仅考虑特定的内存指标，例如托管内存（RocksDB State Backend 等原生进程在 JVM 之外的内存使用情况）	

指标	Unit	描述	Level	使用说明	
managedMemoryUtilization*	百分比	由 managedMemoryUsed/ 派生 managedMemoryTotal	应用程序、运算符、任务、并行性	*仅适用于运行 Flink 版本 1.13 的 KDA 应用程序。 这与 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。	
numberOfFailedCheckpoints	计数	检查点失败的次数。	应用程序	您可以使用此指标来监控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而失败，例如吞吐量或权限问题。	
numRecordsIn	计数	此应用程序、操作员或任务已收到的记录总数。	应用程序、运算符、任务、并行性	该指标的级别指定此指标是衡量整个应用程序、特定操作员或特定任务收到的记录总数。	
numRecordsInPerSecond	计数/秒	此应用程序、操作员或任务每秒收到的记录总数。	应用程序、运算符、任务、并行性	该指标的级别指定此指标是衡量整个应用程序、特定操作员或特定任务每秒收到的记录总数。	
numRecordsOut	计数	此应用程序、操作员或任务发出的记录总数。	应用程序、运算符、任务、并行性	该指标的级别指定该指标是衡量整个应用程序、特定操作员还是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、运算符、任务、并行性		
numRecordsOutPerSecond	计数/秒	此应用程序、操作员或任务每秒发出的记录总数。	应用程序、运算符、任务、并行性	该指标的级别指定此指标是衡量整个应用程序、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收集操作总数。	应用程序		

指标	Unit	描述	Level	使用说明	
oldGenerationGCTime	毫秒	执行旧的垃圾收集操作所花费的总时间。	应用程序	您可以使用此指标来监控总和、平均和最长垃圾收集时间。	
threadCount	计数	应用程序使用的实时线程总数。	应用程序	该指标衡量应用程序代码使用的线程数。这与应用程序并行性不同。	
uptime	毫秒	作业不间断运行的时间。	应用程序	您可以使用此指标来确定作业是否成功运行。对于已完成的任务，此指标返回 -1。	

Kinesis Data Streams

Amazon除以下记录外，还会发出 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（用于直播）、并行性（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（用于直播）、并行性（用于 ShardId）	

Amazon MSK 连接器指标

Amazon除以下记录外，还会发出 Amazon 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）、并行性（用于 PartitionId）	

指标	Unit	描述	Level	使用说明
commitsFailed	不适用	如果启用了偏移量提交和检查点，则向 Kafka 提交偏移量失败的总次数。	应用程序、运算符、任务、并行性	将偏移量提交回 Kafka 只是暴露消费者进度的一种手段，因此提交失败不会影响 Flink checkpoint 分区偏移量的完整性。
commitsSucceeded	不适用	如果启用了偏移量提交和检查点功能，则成功向 Kafka 提交偏移量总数。	应用程序、运算符、任务、并行性	
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）、并行性（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、运算符、任务、并行性	
bytes_consumed_rate	字节	主题平均每秒使用的字节数	应用程序、运算符、任务、并行性	

Apache Zeppelin Metics

对于 Studio 笔记本电脑，在应用程序级别Amazon发出以下指标：KPUscpuUtilization、heapMemoryUtilization、oldGenerationGCTime、oldGenerationGCCount、和threadCount。此外，它还会在应用程序级别发布下表所示的指标。

指标	Unit	描述	Prometheus 名字
zeppelinCpuUtilization	百分比	Apache Zeppelin 服务器中 CPU 利用率的总百分比。	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	等待线程的 Apache Zeppelin 任务的队列数量。	jetty_threads_jobs
zeppelinServerUptime	秒	服务器启动并运行的总时间。	process_uptime_seconds

查看 CloudWatch 指标

您可以使用亚马逊 CloudWatch 控制台或者，查看应用程序的 CloudWatch 指标Amazon CLI。

使用 CloudWatch 控制台查看指标

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
2. 在导航窗格中，选择 Metrics（指标）。
3. 在 Amazon Kinesis Data Analytics 的 “按类别划分的CloudWatch 指标” 窗格中，选择一个指标类别。
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级别。Amazis Data Analytics 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：应用程序报告使用任务指标报告级别定义的指标的任务特定指标维度，例如每秒进出应用程序的记录数。
- 操作员：应用程序报告使用操作员指标报告级别定义的指标的操作员特定的指标维度，例如每个筛选器或地图操作的指标。
- 并行度：应用程序为每个执行线程报告 `Task` 和 `Operator` 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

  由于服务生成的指标数据量很大，因此您只能使用此指标级别进行故障排除。您只能使用 CLI 设置此指标级别。在控制台中不提供此指标级别。

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。

您可以使用操作的参数或CreateApplication操作的MonitoringConfiguration参数来设置 CloudWatch 指标报告级别。MonitoringConfigurationUpdate UpdateApplication以下UpdateApplication操作请求示例将 CloudWatch 指标报告级别设置为 T ask：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "MonitoringConfigurationUpdate": {
            "ConfigurationTypeUpdate": "CUSTOM",
            "MetricsLevelUpdate": "TASK"
         }
```

```
        }
    }
}
```

您也可以使用操作的LogLevel参数或CreateApplication操作的LogLevelUpdate参数配置日志级别。UpdateApplication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 日志级别的更多信息，请参阅 Apache Log4j 文档中的自定义日志级别。

在 Amazon Kinesis Data Analytics 中使用自定义指标

Kinesis Data Analytics for Apache Flink 公开了 19 个指标 CloudWatch，包括资源使用情况和吞吐量指标。此外，您可以创建自己的指标来跟踪应用程序特定的数据，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。Apace 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 指标类型包括计数、仪表、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 Histogram Apache Flink 指标类型。 CloudWatch 只能显示计数、仪表和仪表类型的 Apache Flink 指标。
- 范围：指标的范围由其标识符和一组键值对组成，这些键值对表示将如何向其报告指标 CloudWatch。指标的标识符由以下各项组成：
  - 系统范围，表示报告指标的级别（例如，操作员）。
  - 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MetricGroup.addGroup(key, value)或定义的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 Apache Flink 文档中的指标。

要在适用于 Apache FlAmazon Kinesis Data Analytics 中创建自定义指标，您可以RichFunction通过调用扩展的任何用户函数访问 Apache Flink 指标系统GetMetricGroup。此方法返回一个MetricGroup对象，您可以使用该对象来创建和注册自定义指标。Kinesis Data Analytics 报告使用群组密钥KinesisAnalytics创建的所有指标 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。这些名称必须由字母数字字符组成。
- 您在用户范围（KinesisAnalytics指标组除外）中定义的属性将作为 CloudWatch 维度发布。

- 默认情况下，自定义指标是在该Application级别发布的。
- 根据应用程序的监控级别，将维度（任务/操作员/并行度）添加到指标中。您可以使用操作的参数或CreateApplication操作的或MonitoringConfiguration参数来设置应用程序的UpdateApplication监控级别。MonitoringConfigurationUpdate

示例

以下代码示例演示如何创建映射类来创建和增加自定义指标，以及如何通过将映射类添加到DataStream对象来在应用程序中实现该映射类。

记录数量自定义指标

以下代码示例演示了如何创建映射类，该映射类来创建用于统计数据流中记录的指标（与numRecordsIn指标的功能相同）：

```
    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;
        }
    }
```

在前面的示例中，valueToExpose变量会随着应用程序处理的每条记录而增加。

定义映射类后，然后创建一个实现地图的应用程序内流：

```
DataStream<String> noopMapperFunctionAfterFilter =
    kinesisProcessed.map(new NoOpMapperFunction("FilteredRecords"));
```

有关此应用程序的完整代码，请参阅记录数自定义指标应用程序。

字数自定义指标

以下代码示例演示如何创建映射类来创建用于计算数据流中字数的指标：

```
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));
                    }
                }
            }
        }
```

在前面的示例中，counter变量会随着应用程序处理的每个单词而增加。

定义映射类后，然后创建一个实现地图的应用程序内流：

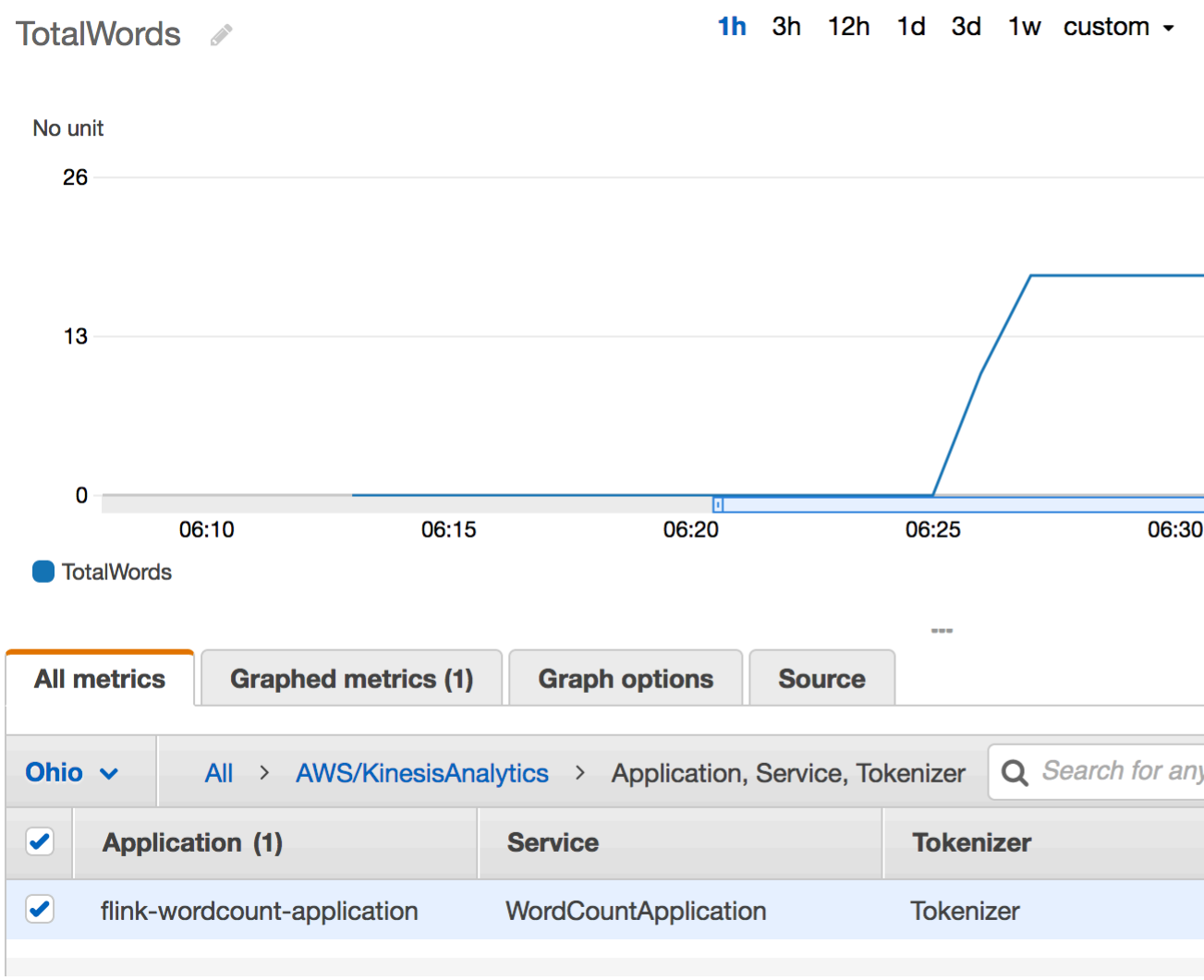
```
// Split up the lines in pairs (2-tuples) containing: (word,1), and
// group by the tuple field "0" and sum up tuple field "1"
DataStream<Tuple2<String, Integer>> wordCountStream = input.flatMap(new
 Tokenizer()).keyBy(0).sum(1);

// 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，请参阅字数自定义指标应用程序。

查看自定义指标

您的应用程序的自定义指标显示在AWS/KinesisAnalytics仪表板的 CloudWatch 指标控制台的应用程序指标组下。



在 Amazon Kinesis Data Analytics 中使用 CloudWatch 警报

使用 Simple Storage Service（Amazon） CloudWatch CloudWatch 指标警报，可以观看指标在指定时间段内的变化。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 CloudWatch 警报的更多信息，请参阅使用亚马逊 CloudWatch 警报。

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。

该表描述了推荐的警报，包括以下几列：

- 指标表达式：根据阈值进行测试的指标或指标表达式。
- 统计数据：用于检查指标的统计数据，例如平均值。

- 阈值：使用此警报要求您确定定义预期应用程序性能限制的阈值。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可能的解决方案。

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 应用程序正在重新启动 (p. 403).
###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

指标表达式	统计数据	Threshold	描述
			troubleshooting, see 检查点已超时 (p. 416).
#### 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. 404).
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 ##_##_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. 404).

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 州无界增长 (p. 405).

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 州无界增长 (p. 405).

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. 33).
`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. 33).
### > 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## 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 CloudWatch 日志

您可以将自定义消息写入 Kinesis Data Analytics 应用程序 CloudWatch 的日志。您可以使用 Apache log4j 库或 Simple Logging Facade for Java (SLF4J) 库以执行该操作。

主题

使用 Log4J 写入 CloudWatch 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 =
 LogManager.getLogger.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 SLF4J 写入 CloudWatch 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 KKinesis Data AnalyticsAmazon CloudTrail

Amazon Kinesis Kinesis Data Analyticticsis Data Analyticticstics Data Analyticticstics 与Amazon CloudTrail集成，后者是一项Amazon提供 Kinesis CloudTrail 将 Kinesis Data Analytics 的所有API 调用作为事件捕获。捕获的调用包括来自 Kinesis Data Analytics 控制台的调用以及对 Kinesis Data Analytics API 操作的代码调 如果您创建跟踪，则可以将 CloudTrail 事件持续传送到 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 Event history（事件历史记录 CloudTrail ）中查看最新事件。通过使用收集的信息 CloudTrail，您可以确定向 Kinesis Data Analytics（分析）、发出请求的 IP 地址、请求发出时间以及其他详细信息。

要了解更多信息 CloudTrail，请参阅Amazon CloudTrail用户指南。

Kinesis Data Analytics CloudTrail

CloudTrail 在您创建Amazon账户时，即针对该账户启用了。当 Kinesis Data Analytics es 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他Amazon服务事件一同保存在 E vent hist ord 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，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录Amazon账户中的事件（包括 Kinesis Data Analytics 的事件），请创建跟踪。通过跟踪 CloudTrail ，可将日志文件传送至 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）桶。此外，您可以配置其他Amazon服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取操作。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
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

所有 Kinesis Data Analytics 操作均由 Kinesis Data Analytics API 参考资料记录 CloudTrail 并记录在这些操作中。例如，对`CreateApplication`和 `UpdateApplication`操作的调用会在 CloudTrail 日志文件中生成条目。

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 日志文件不是公有 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。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明AddApplicationCloudWatchLoggingOption了DescribeApplication操作。


```
{
    "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"
        }
    ]
}
```

在 Apache Flink Data Analytics 中调整性能

本主题介绍监控和改进 Kinesis Data Analytics 应用程序性能的技术。

主题

性能排查的问题

本部分包含症状列表，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较高或不断增加的`MillisBehindLatest`指标。对于其他来源，您可以查看表示从源读取延迟的类似指标。

数据路径

在调查应用程序的性能问题时，请考虑数据采用的整个路径。如果设计或配置不当，以下应用程序组件可能会成为性能瓶颈并产生背压：

- 数据源和目标：确保您的应用程序与之交互的外部资源是针对您的应用程序将要经历的吞吐量预置的。
- 状态数据：确保您的应用程序不会过于频繁地与状态存储交互。

  你可以优化你的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类型，但如果您的应用程序仅以 POJO 类型存储数据，则可以使用性能更高的序列化器。有关 Apache Flink 序列化器的信息，请参阅 Apache Flink 文档中的数据类型和序列化。
- 操作员：确保操作员实现的业务逻辑不太复杂，或者在处理每条记录时不要创建或使用资源。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。

性能故障排除解决方案

本节包含性能问题的潜在解决方案。

主题

- 应用程序内存 (p. 303)

CloudWatch 监控级别

确认 “ CloudWatch 监控级别” 设置未设置为过于详细。

Debug监控日志级别设置会生成大量流量，这可能会产生背压。您只能在积极调查应用程序问题时使用它。

如果您的应用程序Parallelism设置较高，则使用 Parallelism监控指标级别同样会生成大量流量，从而导致背压。仅在应用程序较低或调查应用程序问题时Parallelism使用此指标级别。

有关更多信息，请参阅应用程序监控级别 (p. 274)：

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 Scaling 来允许应用程序为自己分配更多资源。

如果启用了auto 扩展，则如果 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，请参阅以下正确管理扩缩 (p. 303)部分和扩缩 (p. 33)。

> Note
>
> 应用程序只能根据 CPU 使用率自动扩展。应用程序不会根据其他系统指标auto 扩展，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。

应用程序并行性

增加应用程序的并行度。您可以使用UpdateApplication操作的ParallelismConfigurationUpdate参数更新应用程序的并行度。

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。

还必须根据每个操作员的工作负载为其分配并行度，而不仅仅是增加应用程序的并行度。见操作员并行性 (p. 302)下文。

应用程序日志

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。有关读取应用程序日志的更多信息，请参阅the section called “分析日志” (p. 275)。

操作员并行性

验证您的应用程序的工作负载是否在工作进程之间均匀分布。

有关调整应用程序操作员工作负载的信息，请参阅操作员扩展 (p. 303)。

应用程序逻辑

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高或无法可靠访问，它也会影响性能，这可能会导致外部依赖项返回HTTP 500错误。

如果您的应用程序使用外部依赖关系来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 A pache Flink 文档中的异步 I/O。

应用程序内存

检查您的应用程序是否存在资源泄漏。如果您的应用程序未正确处置线程或内存，则可能会看到`MillisBehindLatestCheckpointSize`、和`CheckpointDuration`指标激增或逐渐增加。这种情况还可能导致任务管理器或任务管理器故障。

性能最佳实践

本节介绍为提高性能而设计应用程序时的特殊注意事项。

正确管理扩缩

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展 (p. 303)
- 正确管理操作员扩展 (p. 303)

正确管理应用程序扩展

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。
- CPU 使用率在 15 分钟内保持在 75% 以上。

如果启用了自动缩放，但 CPU 使用率未保持在此阈值，则应用程序将无法扩展 KPU。如果您遇到 CPU 使用率峰值未达到此阈值，或者遇到其他使用率指标（例如）的峰值`heapMemoryUtilization`，请手动增加扩展能力，让您的应用程序能够应对活动峰值。

> Note
>
> 如果应用程序通过 Auto Scaling 自动添加了更多资源，则该应用程序将在一段时间不活动后释放新资源。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. 33)。

正确管理操作员扩展

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布，以及应用程序中的操作员是否拥有保持稳定和高性能所需的系统资源，来提高应用程序的性能。

您可以使用设置为应用程序代码中的每个运算符`parallelism`设置并行度。如果您不为运算符设置并行度，它将使用应用程序级并行度设置。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的所有可用系统资源，从而使应用程序变得不稳定。

为了最好地确定每个运算符的并行度，请考虑该运算符与应用程序中其他运算符相比的相对资源需求。将资源密集度更高的运算符设置为比资源密集度较低的运算符更高的运算符并行度设置。

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可用于应用程序的任务槽总数之间的最佳比率来调整应用程序的总操作员并行度。操作员总并行度与任务槽的典型稳定比率为 4:1，也就是说，应用程序每四个可用的操作员子任务就有一个任务槽可用。具有更多资源密集型运算符的应用程序可能需要的比率为 3:1 或 2:1，而具有资源密集度较低的运算符的应用程序可能以 10:1 的比率保持稳定。

您可以使用设置运算符的比率运行时属性 (p. 25)，因此无需编译和上传应用程序代码即可调整运算符的并行度。

下面的代码示例演示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，请参阅应用程序资源 (p. 7)。

要控制应用程序工作进程之间工作负载的分配，请使用Parallelism设置和KeyBy分区方法。有关更多信息，请参阅 Apache Flink 文档中的以下主题：

- 并行执行
- DataStream 转换

监控外部依赖资源使用情况

如果目的地（例如 Kinesis Streams、Kinesis Data Firehose、DynamoDB 或 OpenSearch 服务）存在性能瓶颈，您的应用程序将面临背压。验证是否已为应用程序吞吐量正确配置了外部依赖项。

Note

其他服务中的故障可能会导致您的应用程序出现故障。如果您发现应用程序出现故障，请检查目标服务的 CloudWatch 日志中是否存在故障。

在本地运行你的 Apache Flink 应用程序

要解决内存问题，你可以在本地 Flink 安装中运行应用程序。这将允许您访问调试工具，例如堆栈跟踪和堆转储，这些工具在 Kinesis Data Analytics 中运行应用程序时不可用。

有关创建本地 Flink 安装的信息，请参阅 Apache Flink 文档中的本地安装教程。

监控性能

本节介绍用于监控应用程序性能的工具。

使用 CloudWatch 指标进行性能监控

您可以使用 CloudWatch 指标监控应用程序的资源使用情况、吞吐量、检查点和停机时间。有关在 Kinesis Data Analyics 应用程序中使用 CloudWatch 指标的信息，请参阅指标与维度 (p. 278)。

使用 CloudWatch 日志和警报进行性能监控

您可以使用 CloudWatch 日志监控可能导致性能问题的错误情况。

当 Apache Flink 作业状态从状态变为状态时，错误条件会RUNNING出现在日志条目中FAILED。

您可以使用 CloudWatch 警报为性能问题创建通知，例如资源使用情况或检查点指标超过安全阈值或应用程序状态意外变化。

有关为 Kinesis Data Analyics 应用程序创建 CloudWatch 警报的信息，请参阅告警 (p. 290)。

Kinesis Data Analytics for Apache Flink、

使用Apache FAmazon Kinesis Data Analytics
k、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、

- 您可以为账户中的每个区域创建最多 50 个 Kinesis Data Analytics 应用程序。您可以创建一个案例，通过服务配额增加表单请求其他申请。有关更多信息，请参阅该Amazon Web Services Support中心。

  有关支持 Kinesis Data Analytics 的区域列表，请参阅 Kinesis Data Analytics 区域和端点。

- 默认情况下，Kinesis 处理单元 (KPU) 的数量限制为 32。有关如何申请增加此配额的说明，请参阅申请增加Service Quota s。确保指定需要应用新 KPU 限制的应用程序前缀。

  使用 Kinesis Data Analytics，您的Amazon账户需要为分配的资源付费，而不是按应用程序使用的资源付费。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。单个 KPU 可为您提供 1 个 vCPU 和 4 GiB 内存。对于每个 KPU，该服务还提供 50 GiB 的运行应用程序存储。

- 快照 (p. 30)每个应用程序最多可以创建 1,000 个 Kinesis Data Analytics。

- 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。如果您超出此配额，您的应用程序将无法启动。

对于 Studio 笔记本电脑，以下配额将适用。要申请更高的配额，请创建支持案例。

- `websocketMessageSize`= 5 MiB
- `noteSize`= 5 MiB
- `noteCount`1000、、、
- `Max cumulative UDF size`= 100 MiB
- `Max cumulative dependency jar size`= 300 MiB

维护Kinesis Data Analytics 维护维护维护维护维护维护维护维护维护维护维护维护维护维护维护维护

Kinesis Data Analytics 使用操作系统和容器映像安全更新定期修补您的应用程序，以保持合规性并实现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。您的应用程序维护可能在与您所在地区对应的时间窗口内的任何时候进行。在此维护过程中，您的应用程序可能会出现 10 到 30 秒的停机时间。但是，实际的停机时间取决于应用程序状态。有关如何最大限度地减少停机时间影响的信息，请参阅the section called “容错：检查点和保存点” (p. 310)。

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	维护维护维护段维护段维护
Amazon GovCloud （美国西部）	06:00–14:00 UTC
Amazon GovCloud （美国东部）	03:00–11:00 UTC
美国东部（弗吉尼亚州北部）	03:00–11:00 UTC
美国东部（俄亥俄州）	03:00–11:00 UTC
美国西部（北加利福尼亚）	06:00–14:00 UTC
美国西部（俄勒冈州）	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（海得拉巴）	06:30–14:30 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（雅加达）	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（苏黎世）	02:00–10:00 UTC

区域	维护维护维护段维护段维护
欧洲（爱尔兰）	22:00–06:00 UTC
欧洲（伦敦）	22:00–06:00 UTC
欧洲（斯德哥尔摩）	21:00–05:00 UTC
欧洲（米兰）	21:00–05:00 UTC
欧洲（西班牙）	02:00–10:00 UTC
非洲（开普敦）	13:00–21:00 UTC
欧洲（爱尔兰）	22:00–06:00 UTC
欧洲（伦敦）	23:00–07:00 UTC
欧洲（巴黎）	23:00–07:00 UTC
欧洲（斯德哥尔摩）	23:00–07:00 UTC
中东（巴林）	13:00–21:00 UTC
南美洲（圣保罗）	世界标准时间 19:00 — 03:00
中东（阿联酋）	13:00–21:00 UTC

为所有操作员设置 UUID

当 Kinesis Data Analytics 为带有快照的应用程序启动 Flink 作业时，Flink 作业可能由于某些问题而无法启动。其中之一是操作员 ID 不匹配。Flink 期望 Flink 任务图运算符具有明确、一致的运算符 ID。如果没有明确设置，Flink 会自动为运算符生成 ID。这是因为 Flink 使用这些运算符 ID 来唯一识别任务图中的运算符，并使用它们将每个运算符的状态存储在保存点中。

当 Flink 找不到任务图的运算符 ID 和保存点中定义的运算符 ID 之间的 1:1 映射时，就会出现运算符 ID 不匹配问题。当未设置明确的一致运算符 ID 并且 Flink 自动生成可能与每次创建任务图时不一致的运算符 ID 时，就会发生这种情况。在维护运行期间，应用程序遇到此问题的可能性很高。为了避免这种情况，我们建议客户在 flink 代码中为所有运算符设置 UUID。有关更多信息，请参阅为处于生产就绪状态的所有操作员设置 UUID 主题。

生产准备就绪

这是在 Kinesis Data Analytics 上运行生产应用程序的重要方面的集合。这不是一份详尽的清单，而是在将应用程序投入生产之前应注意的最低限度的内容。

加载加载加载加载加载加载加载

应用程序的某些问题仅在高负载下才会出现。我们见过这样的情况：应用程序看起来很健康，操作事件大大增加了应用程序的负载。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 应用程序将无法取得进展。一旦该问题得到解决，就会积压大量未处理的数据，这可能会完全耗尽可用资源。然后，负载会放大以前未出现的错误或性能问题。

因此，必须对生产应用程序进行适当的负载测试。在这些负载测试期间应回答的问题包括：

- 应用程序在持续的高负载下是否稳定？
- 在峰值负载下，应用程序还能保存点吗？
- 处理来自 1 小时的结果需要多长时间？ 以及 24 小时需要多长时间（取决于流中数据的最大保留时间）？
- 扩展应用程序时，应用程序的吞吐量是否会增加？

当从数据流中消耗时，可以通过在数据流中生成一段时间来模拟这些场景。然后启动应用程序，让它从一开始就消耗数据，例如，对于 Kinesis Data Stream，使用起始位置。`TRIM_HORIZON`

为 DAG 中的每个运算符设置明确的最大并行度

这应该是你对应用程序`total parallelism`的函数。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)
```

其中`MAX_PARALLELISM`，是一个等于结果的变量`env.getParallelism()`。

为所有操作员设置 UUID

在 Flink 将保存点映射回单个运算符的操作中使用了 UUID。为每个操作员设置特定的 UUID 可以为要恢复的保存点进程提供稳定的映射。

```
.map(...).uid("my-map-function")
```

有关更多信息，请参阅生产测试清单

适用于 Apache Flink

本部分包含开发稳定、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 应用程序中实现容错能力。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。Checkpointing 可在计划维护期间以及服务问题、应用程序依赖关系故障和其他问题导致意外故障时为您的应用程序提供容错能力。有关定期维护的信息，请参见维护 (p. 307)。
- `false`在应用程序开发或故障排除期间SnapshotsEnabled将ApplicationSnapshotConfiguration:: 设置为。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

  我们建议您的应用程序每天创建几次快照，以便使用正确的状态数据正确重启。快照的正确频率取决于应用程序的业务逻辑。频繁拍摄快照可以恢复更新的数据，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. 278)。

有关实施容错功能的更多信息，请参阅容错能力 (p. 27)。

不支持的连接器版本

如果应用程序使用不支持的 Kinesis Connector 版本（捆绑到应用程序 JAR 中），则适用于 Apache Flink 的 Kinesis Data Analytics 版本 1.15 将自动阻止应用程序启动或更新。升级到适用于 Apache Flink 的 Kinesis

Data Analytics 版本 1.15 时，请确保你使用的是最新的 Kinesis Connector。这是任何等于或高于 1.15.2 的版本。适用于 Apache Flink 的 Kinesis Data Analytics 将不支持所有其他版本，因为它们可能会导致一致性问题或故障，而 Stop with Savepoint 功能会阻止干净的停止 /更新操作。

性能和并行性

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度的更多信息，请参阅明确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. 33)。

日志记录

您可以使用 CloudWatch 日志监控应用程序的性能和错误状况。为应用程序配置日志记录时，请牢记以下几点：

- 启用应用程序 CloudWatch 的日志记录，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。
- 创建 CloudWatch 警报以在应用程序无法正常运行时通知您。有关更多信息，请参阅 告警 (p. 290)

有关实施日志记录的更多信息，请参阅日志记录和监控 (p. 268)。

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：
  - 如果您的应用程序的`main`方法引发了异常，则当应用程序过渡到`RUNNING`状态`ProgramInvocationException`时，服务会将其封装在 a 中，任务管理器将无法提交作业。
  - 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

您不应在生产应用程序（或任何其他）应用程序中使用任何长期证书。长期凭证很可能会被存入版本控制系统，很容易丢失。相反，您可以将角色关联到 Kinesis Data Analytics 应用程序并为该角色授予权限。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 原生集成的服务（例如需要用户名和密码进行身份验证的数据库）需要身份验证，则应考虑在 Secrets Manager 中存储Amazon机密。

许多Amazon原生服务支持身份验证：

- Kinesis Data Streams — ProcessTaxiStream.java
- 亚马逊 MSK — https://github.com/aws/aws-msk-iam-auth /#using-the-amazon-msk-library-for-iam-authentication
- Amazon Elasticsearch Servic AmazonElasticsearchSinke — .java
- Amazon S3 — 在 Kinesis Data Analytics 上开箱即用

从分片/分区很少的源代码中读取

从 Apache Kafka 或 Kinesis Data Stream 读取数据时，数据流的并行度（即 Kafka 的分区数和 Kinesis 的分区数）与应用程序的并行度之间可能不匹配。在天真的设计中，应用程序的并行性无法扩展到流的并行性之外：源运算符的每个子任务只能从 1 个或多个分片/分区读取。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在消耗流中，而且 6 个子任务处于空闲状态。这可能会极大地限制应用程序的吞吐量，尤其是在反序列化成本高昂且由源代码执行（这是默认设置）的情况下。

为了减轻这种影响，您可以缩放直播。但这可能并不总是可取或可能的。或者，你可以重构源代码，使其不进行任何序列化，只传递`byte[]`。然后，您可以重新平衡数据，将其均匀地分布在所有任务中，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，并且这种潜在的昂贵操作不再受流分片/分区数量的限制。

工作室笔记本刷新间隔

如果更改段落结果刷新间隔，将其设置为至少为 1000 毫秒的值。

Studio 笔记本电脑的最佳性能

我们使用以下语句进行了测试，`events-per-second`乘以小`number-of-keys`于 25,000,000 时获得了最佳性能。这是为`events-per-second`不到15万的人准备的。

```
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

读取来自 Apache Kafka 和 Kinesis Data Streams 的事件时，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，将有关事件时间的信息从源传播给所有其他操作员。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 Kinesis 确定的大致到达时间。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，DataStream使用该assignTimestampsAndWatermarks方法将水印策略应用于。有一些有用的内置策略：

- forMonotonousTimestamps()将只使用事件时间（大概到达时间）并定期发出最大值作为水印（针对每个特定的子任务）
- forBoundedOutOfOrderness(Duration.ofSeconds(...))与之前的策略类似，但会使用事件时间，即生成水印的持续时间。

这行得通，但有几个注意事项。水印是在子任务级别生成的，并流经运算符图。

来自 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。

当水印流过直播节目时，它们会提前到达运营商处的活动时间。每当操作员提前其事件时间时，它都会为其后续操作员在下游生成一个新的水印。

有些运算符会消耗多个输入流；例如，联合运算符或 keyBy (...) 或分区 (...) 函数之后的运算符。此类操作员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，如果源子任务消耗来自空闲分片，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的处理都会停止。为避免这种情况，客户可以在水印策略中添加该withIdleness选项。使用该选项，操作员在计算操作员的事件时间时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操作员的事件时间延长。

但是，如果没有子任务读取任何事件，即流中没有事件，则带有内置水印策略的空闲选项不会延长事件时间。这对于从流中读取一组有限事件的测试用例来说尤其明显。由于读取最后一个事件后事件时间没有提前，因此最后一个窗口（包含最后一个事件）永远不会关闭。

摘要

- 如果分区处于空闲状态，该withIdleness设置不会生成新的水印，它只会从下游运算符的最小水印计算中排除空闲子任务发送的最后一个水印
- 使用内置水印策略，上次打开的窗口永远不会关闭（除非会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 流设定，如果一个分片的消耗速度比其他分片快（例如，在应用程序初始化期间，或者TRIM_HORIZON在使用parallel 消耗所有现有分片时忽略它们的父/子关系），延迟到达的事件仍会发生
- 水印策略的withIdleness设置似乎不推荐使用空闲分片的 Kinesis 源代码特定设置(ConsumerConfigConstants.SHARD_IDLE_INTERVAL_MILLIS

示例

以下应用程序正在读取直播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 个分区流（前 2 个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": "496278943385700540701037497830421167324199343939366422 10"
}
{
    "ShardId": "shardId-000000000014",
    "SequenceNumber": "496278943386592570508978722756590344899343423340177000 66"
}
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:
 148873535527910577652263907513985925 12,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:
 319014718988379809496913694467282698240,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:
 496278943384362495989125660432414778027473288653837435554,}}'} 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:
 63802943797675961899382738893456539648,EndingHashKey:
 85070591730234615865843651857942052863},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:
 21267647932558653966460912964485513216,EndingHashKey:
 42535295865117307932921825928971026431},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:
 19140883139302788569814821668036961894 4,EndingHashKey:
 21267647932558653966460912964485513215 9},SequenceNumberRange: {StartingSequenceNumber:
 496278943385477533249052191589491563941105706729136457 14,}}'}, 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:
 21267647932558653966460912964485513215},SequenceNumberRange: {StartingSequenceNumber:
 496278943383470466181184435506753349296567354193598218 26,}}'}, 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:
 25521177519070384759753095557382615859 2,EndingHashKey:
 27647942312326250156399186853831167180 7},SequenceNumberRange: {StartingSequenceNumber:
 496278943386146555605008110283737635489285157574315870 10,}}'}, 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:
 17014118346046923173168730371588410 5728,EndingHashKey:
 19140883139302788569814821668036961894 3},SequenceNumberRange: {StartingSequenceNumber:
 496278943385254525797066885358076206758379223114076652 82,}}'}, 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:
 21267647932558653966460912964485513216,EndingHashKey:
 42535295865117307932921825928971026431},SequenceNumberRange: {StartingSequenceNumber:
 496278943383693473633169741738168706479293837808658022 58,}}'} 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:
 23394412725814519363107004260934064537 6,EndingHashKey:
 25521177519070384759753095557382615859 1},SequenceNumberRange: {StartingSequenceNumber:
 496278943385923548153022804052322278306558673959256065 78,}}'}, 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:
 21267647932558653966460912964485513215},SequenceNumberRange: {StartingSequenceNumber:
```

```
 496278943383470466181184435506753349296567354193598218 26,}}'} 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:
 42535295865117307932921825928971026432,EndingHashKey:
 63802943797675961899382738893456539647},SequenceNumberRange: {StartingSequenceNumber:
 496278943383916481085155047969584063662020321423717826 90,}}'}, 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:
 27647942312326250156399186853831167 1808,EndingHashKey:
 29774707105582115553045278150279718 5023},SequenceNumberRange: {StartingSequenceNumber:
 496278943386369563056993416515152992672011641189375674 42,}}'}, 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:
 496278943383916481085155047969584063662020321423717826 90,}}'} 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:
 19140883139302788569814821668036961894 4,EndingHashKey:
 21267647932558653966460912964485513215 9},SequenceNumberRange: {StartingSequenceNumber:
 496278943385477533249052191589491563941105706729136457 14,}}'} 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:
 21267647932558653966460912964485513216 0,EndingHashKey:
 23394412725814519363107004260934064537 5},SequenceNumberRange: {StartingSequenceNumber:
 496278943385700540701037497820906921123832190344196261 46,}}'} 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:
 31901471898837980949691369446728269824 0,EndingHashKey:
 34028236692093846346337460743176821145 5},SequenceNumberRange: {StartingSequenceNumber:
 496278943386815577960964028977983707037464608419495283 06,}}'} 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:
 29774707105582115553045278150279718 5024,EndingHashKey:
 31901471898837980949691369446728269823 9},SequenceNumberRange: {StartingSequenceNumber:
 496278943386592570508978722746568349854738124804435478 74,}}'} 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:
 17014118346046923173168730371588410572 8,EndingHashKey:
 19140883139302788569814821668036961894 3},SequenceNumberRange: {StartingSequenceNumber:
 496278943385254525797066885358076206758379223114076652 82,}}'} 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:
 27647942312326250156399186853831167 1808,EndingHashKey:
 29774707105582115553045278150279718 5023},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:
 25521177519070384759753095557382615 8592,EndingHashKey:
 27647942312326250156399186853831167 1807},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:
 23394412725814519363107004260934064537 6,EndingHashKey:
 25521177519070384759753095557382615 8591},SequenceNumberRange: {StartingSequenceNumber:
 49627894338592354815302280405232227830655867395925606578,}}'} 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
```

输出仅显示 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，因为使用预先构建的水印策略，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，时间需要在上次事件之后延长 10 秒以上。在本例中，最后一个水印是 2022-03-23T10：21:27 .170Z，但要关闭会话窗口，需要 10 秒和 1 毫秒后的水印。

如果将该`withIdleness`选项从水印策略中删除，则不会关闭任何会话窗口，因为窗口操作员的 “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，而没有消耗其订阅的其他分片的水印）。缓解方法是不同的水印策略，这些策略会增加安全缓冲区(`forBoundedOutOfOrderness(Duration.ofSeconds(30))`或明确允许延迟到来的事件(`allowedLateness(Time.minutes(5))`。

为所有操作员设置 UUID

当 Kinesis Data Analytics 为带有快照的应用程序启动 Flink 作业时，Flink 作业可能由于某些问题而无法启动。其中之一是操作员 ID 不匹配。Flink 期望 Flink 任务图运算符具有明确、一致的运算符 ID。如果没有明

确设置，Flink 会自动为运算符生成 ID。这是因为 Flink 使用这些运算符 ID 来唯一识别任务图中的运算符，并使用它们将每个运算符的状态存储在保存点中。

当 Flink 找不到任务图的运算符 ID 和保存点中定义的运算符 ID 之间的 1:1 映射时，就会出现运算符 ID 不匹配问题。当未设置明确的一致运算符 ID 并且 Flink 自动生成可能与每次创建任务图时不一致的运算符 ID 时，就会发生这种情况。在维护运行期间，应用程序遇到此问题的可能性很高。为了避免这种情况，我们建议客户在 flink 代码中为所有运算符设置 UUID。有关更多信息，请参阅为处于生产就绪状态的所有操作员设置 UUID 主题。

Apache Flink 有状态函数

Stateful Functions 是一种 API，可简化分布式有状态应用程序的构建。它基于具有持久状态的函数，这些函数可以在强大的一致性保证下动态交互。

Stateful Functions 应用程序基本上只是一个 Apache Flink 应用程序，因此可以部署到 Kinesis Data Analytics 中。但是，为 Kubernetes 集群打包有状态函数和为 Kinesis Data Analytics 打包有状态函数之间有一些区别。Stateful Functions 应用程序最重要的方面是模块配置包含配置 Stateful Functions 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 Kinesis Data Anal StateFun ytics 的 Python 示例的改编：

Apache Flink 应用程序模板

客户无需使用客户容器来运行 Stateful Functions 运行时，而是可以编译 Flink 应用程序 jar，该jar仅调用 Stateful Functions 运行时并包含所需的依赖关系。对于 Flink 1.13，所需的依赖关系如下所示：

```
<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>
```

而 Flink 应用程序调用 Stateful Function 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 Stateful 函数中实现的逻辑。

对模块配置的位置

Stateful Functions 模块配置需要包含在类路径中才能被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，你可以使用 maven 创建 uber jar 文件并将其部署在 Kinesis Data Analytics 上。

Flink Kinesis Flink

本主题包含有关在旧版本的 Apache Flink 上使用 Kinesis Data Analytics 的信息。Kinesis Data Analytics 支持的 Apache Flink 版本为 1.15.2（推荐）、1.13.2、1.11.1、1.8.2 和 1.6.2。

建议您使用支持的最新版本 Apacace FKinesis Data Analytics。Apache Flink 版本 1.15.2 具有以下功能：

- Support Apache Flink Table API 和 SQL
- Support 的 Pyton Pytics
- Support Java 版本 11 和 Scala 版本 2.12
- 改进的内存模型
- RocksDB 优化可提高应用程序稳定性
- 在 Apache Flink 控制面板中Support 任务管理器和堆栈跟踪。

本主题包含下列部分：

在之前的 Apache Flink Kinesis Streams 连接器中使用 Apache Flink

在 1.11 版本之前，Apache Flink Kinesis Streams 连接器不包含在 Apache Flink 中。为了让您的应用程序将 Apache Flink Kinesis 连接器与 Apache Flink 的先前版本一起使用，您必须下载、编译和安装您的应用程序使用的 Apache Flink 版本。此连接器用于处理用作应用程序源的 Kinesis 流中的数据，或将数据写入用于应用程序输出的 Kinesis 流。

> Note
>
> 确保使用 KPL 版本 0.14.0 或更高版本构建连接器。

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

如果你在微软 Windows 上编译 Flink，你需要添加`-Drat.skip=true`参数。

使用 Apacacle Flink Flink 1.8.2

本节包含有关用于构建与 Apache Flink 1.8.2 配合使用的 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
```

有关使用 Apache Flink 版本 1.8.2 的 Kinesis Data Analytics 应用程序`pom.xml`的文件示例，请参阅 Fl ink 1.8.2 版 Kinesis Data Analytics 入门应用程序。

有关如何为 Kinesis 应用程序，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 配合使用的 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
Amazon Beat	Apacle Flink Flink 1.6.2。

Note

使用 Kinesis Data Analytics 运行时版本 1.0.1 时，您可以在`pom.xml`文件中指定 Apache Flink 的版本，而不是在编译应用程序代码时使用`-Dflink.version`参数。

有关使用 Apache Flink 版本 1.6.2 的 Kinesis Data Analytics 应用程序`pom.xml`的文件示例，请参阅 Fl ink 1.6.2 版 Kinesis Data Analytics 入门应用程序。

有关如何为 Kinesis 应用程序，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，必须更新应用程序代码，删除以前的应用程序，并使用更新后的代码创建新应用程序。为此，请执行以下操作：

- 将应用程序`pom.xml`文件中的 Kinesis Data Analytics 运行时和 Kinesis Data Analytics Flink 连接器 (aws-kinesisanalytics-flink) 的版本更改为 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=1.15.3
```

- 删除现有的应用程序。再次创建您的应用程序，然后选择 Apache Flink 版本 1.15.2（推荐版本）作为应用程序的运行时间。

Note

您不能使用以前应用程序版本的快照。

Apacacle Flink Flink 1.6.2 和 1.8.2

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 框架中可用连接器的信息，请参阅 Apache Flink 文档 (1.6.2) 中的连接器 (1.6.2)。
- 有关 Apache Flink 1.8.2 框架中可用连接器的信息，请参阅 Apache Flink 文档 (1.8.2) 中的连接器 (1.8.2)。

入门：Flink 1.13.2

本节向你介绍适用于 Apache Flink 和 DataStream API 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

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 运行时处理输入并生成输出。

Kinesis Data Analytics 应用程序包含以下组件：

- 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：应用程序通过源 使用数据。源连接器从 Kinesis 数据流、Amazon S3 存储桶等读取数据。有关更多信息，请参阅源 (p. 14)：
- 操作符：应用程序使用一个或多个操作符 以处理数据。操作符可以转换、丰富或聚合数据。有关更多信息，请参阅DataStream API 操作员 (p. 18)：
- 接收器：应用程序使用接收器 将生成的数据发送到外部源。Sink 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. 15)：

创建、编译和打包应用程序代码后，您将代码包上载到 Amazon Simple Storage Service (Amazon Simple Storage Service) 桶。然后，您创建 Kinesis Data Analytics 应用程序。您传入代码包位置、作为流数据源的 Kinesis 数据流，通常传入接收应用程序处理过的数据流或文件位置。

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，请转到步骤 1：设置 Amazon 账户并创建管理员用户 (p. 88)。

步骤 1：设置 Amazon 账户并创建管理员用户

注册一个 Amazon Web Services 账户

如果您还没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

注册 Amazon Web Services 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。

   当您注册 Amazon Web Services 账户 时，系统将会创建一个 Amazon Web Services 账户根用户。根用户有权访问该账户中的所有 Amazon Web Services和资源。作为安全最佳实践，请 为管理用户分配管理访问权限，并且只使用根用户执行 需要根用户访问权限的任务。

Amazon注册过程完成后，会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择 My Account (我的账户) 来查看当前的账户活动并管理您的账户。

保护 IAM 用户

注册 Amazon Web Services 账户 后，启用多重身份验证（MFA）保护您的管理用户。有关说明，请参阅 IAM 用户指南中的为 为 IAM 用户（控制台）启用虚拟 MFA 设备。

要授予其他用户访问您的 Amazon Web Services 账户 资源的权限，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA 并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅 IAM 用户指南中的以下主题：

- 在您的 Amazon Web Services 账户 中创建 IAM 用户
- 适用于 Amazon 资源的访问管理
- IAM 基于身份的策略示例

授权以编程方式访问

如果用户需要在 Amazon Web Services Management Console 之外与 Amazon 交互，则需要编程式访问权限。Amazon API 和 Amazon Command Line Interface 需要访问密钥。可能的话，创建临时凭证，该凭证由一个访问密钥 ID、一个秘密访问密钥和一个指示凭证何时到期的安全令牌组成。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
IAM	使用短期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中将临时凭证用于 Amazon 资源中的说明进行操作。

哪个用户需要编程式访问权限？	目的	方式
IAM	（不推荐使用） 使用长期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中管理 IAM 用户的访问密钥中的说明进行操作。

下一个步骤

步骤 2：设置 Amazon Command Line Interface (Amazon CLI) (p. 328)

下一个步骤

步骤 2：设置 Amazon Command Line Interface (Amazon CLI) (p. 328)

步骤 2：设置 Amazon Command Line Interface (Amazon CLI)

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

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《Amazon Command Line Interface用户指南》Amazon Command Line Interface中的“安装”。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。有关命名配置文件的更多信息，请参阅Amazon Command Line Interface用户指南中的命名配置文件。

   ```
   [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Amazon区域的列表，请参阅《Amazon Web Services 一般参考》中的区域和终端节点。

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
```

在您设置Amazon帐户后Amazon CLI，您可以尝试下一个练习，在其中配置示例应用程序并测试 end-to-end 设置。

下一个步骤

步骤 3：为 Apache Flink 应用程序创建并运行 Kinesis Data Analytics (p. 329)

步骤 3：为 Apache Flink 应用程序创建并运行 Kinesis Data Analytics

在本练习中，您将创建一个以数据流作为源和汇点的 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 应用程序之前，请创建两个 Kinesis 数据流（`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Amazon CLI命令创建这些直播。有关控制台的说明，请参阅 Amazon Kinesis 数据流开发者指南中的创建和更新数据流。

创建数据流 (Amazon CLI)

1. 要创建第一个直播 (`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', region_name='us-west-2'))
```

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：

- 项目对象模型 (pom.xml) 文件包含有关应用程序配置和依赖关系的信息，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 源从源码流中读取数据。以下片段创建了 Kinesis 源代码：

```
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. 25)。

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. 87)。

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

在本部分中，您创建亚马逊简单存储服务 (Amazon Simple Storage Service (Amazon Simple Service)

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. **`ka-app-code-<username>`**在存储段名称字段中输入。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. 选择创建桶。
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。选择下一步。
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存储在 Amazon S3 存储桶中，应用程序可以在其中访问。

创建并运行 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，会为您创建Amazon Identity and Access Management (IAM) 和 A CloudWatch mazon Logs 资源。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，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。
   - 将版本下拉列表保留为 Apache Flink 版本 1.13。
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 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",
                "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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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. 输入以下信息:

组 ID	键	值
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

5. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
6. 要进行CloudWatch 记录，请选中 “启用” 复选框。
7. 选择更新。

> Note
>
> 当您选择启用亚马逊 CloudWatch 日志记录时，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 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

停止应用程序

在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，也可以从 Amazon S3 存储桶重新加载应用程序 JAR。

在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)

在本节中，您将使用创建和运行 Kinesis Data Analytics 应用程序。Amazon CLI适用于 Apache Flink 的 Kinesis Data Analytics 使用该`kinesisanalyticsv2`Amazon CLI命令创建 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（您在下一节中创建）。因此，当 Kinesis Data Analytics 担任该角色时，该服务具有从源流读取和写入接收流的必要权限。

使用以下代码创建 KAReadSourceStreamWriteSinkStream 权限策略。*username*替换为用于创建 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"
        }
    ]
}
```

有关创建权限策略的 step-by-step 说明，请参阅 IAM 用户指南中的教程：创建并附加您的第一个客户托管策略。

> **Note**
>
> 要访问其他亚马逊服务，您可以使用Amazon SDK for Java。Kinesis Data Analytics 会自动将软件开发工具包所需的证书设置为与您的应用程序关联的服务执行 IAM 角色的证书。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 应用程序可以假定该角色来读取源流并写入接收流。

未经许可，Kinesis Data Analytics 无法访问您的直播。您可以通过 IAM 角色授予这些权限。每个 IAM 角色都附加了两个策略。信任策略允许 Kinesis Data Analytics 代入角色，权限策略决定了 Kinesis Data Analytics 在代入角色后可以做什么。

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 “选择可信身份类型” 下，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 Attach permissions policies 页面上，选择 Next: Review。在创建角色后，您可以附加权限策略。
5. 在创建角色页面上**`KA-stream-rw-role`**，输入角色名称。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 Analytics 扮演这个角色，既可以从 Kinesis 数据流（源）读取数据，也可以将输出写入另一个 Kinesis 数据流。因此，您附加在上一步（the section called “创建权限策略” (p. 335)）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。

有关创建角色的 step-by-step 说明，请参阅 IAM 用户指南中的创建 IAM 角色（控制台）。

创建 Kinesis Data Analytics

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_15",
    "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 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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 日志选项

您可以使用Amazon CLI向您的应用程序添加 Amazon CloudWatch 日志流。有关在应用程序中使用 CloudWatch Logs 的信息，请参阅the section called “设置日志记录” (p. 269)。

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。在此示例中，您更改了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

要加载具有相同文件名的新版本的应用程序代码，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。

要使用Amazon CLI，请从 Amazon S3 存储桶中删除您之前的代码包，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. 90)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

步骤 4：清除Amazon资源 (p. 339)

步骤 4：清除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 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，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream。
3. 在ExampleInputStream页面中，选择 “删除 Kinesis Stream”，然后确认删除。

4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择 “操作”，选择 “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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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

步骤 5：后续步骤 (p. 340)

步骤 5：后续步骤

现在，您已经创建并运行了基本的 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。

- Amazon Kinesis 的Amazon流媒体数据解决方案：Amazon Kinesis 的流媒体数据解决方案可自动配置必要的Amazon服务，以便轻松捕获、存储、处理和传输流数据。Amazon该解决方案为解决流数据用例提供了多个选项。Kinesis Data Analytics 选项提供了一个 end-to-end流式传输 ETL 示例，演示了对模拟的纽约出租车数据进行分析操作的真实应用程序。该解决方案设置了所有必需的Amazon资源，例如 IAM 角色和策略、 CloudWatch 仪表板和 CloudWatch 警报。
- AmazonAmazon MSK 的Amazon流媒体数据解决方案：Amazon MSK 的流媒体数据解决方案提供了数据流经生产者、流媒体存储、消费者和目的地的Amazon CloudFormation模板。
- 带有 Apache Flink 和 Apache Kafka 的 Clickst ream Lab：一个端到端的点击流实验室，使用适用于 AManaged Streaming for Apache Kafka 的 Amazon Managed Streaming 进行流存储，使用适用于 Apache Flink 的Amazon Kinesis Data Analytics 应用程序进行流处理。
- 流媒体分析研讨会：在本研讨会中，您将构建一个 end-to-end 流媒体架构，以近乎实时地提取、分析和可视化流媒体数据。你着手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队的遥测数据，以优化其车队运营。

- 适用于 Apach Flink (p. 146)：本开发者指南的这一部分提供了在 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，可帮助您创建 Kinesis Data Analytics 应用程序和测试结果。
- 学习 Flink：动手训练：Apache Flink 官方入门培训，让你开始编写可扩展的流媒体 ETL、分析和事件驱动应用程序。

  Note

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

入门指南：：：：：：：：：：：：：：

本主题包含使用 Apache Flink 1.11.1 的入门指南 (DataStream API) (p. 87)教程版本。

本节向你介绍适用于 Apache Flink 和 DataStream API 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

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 运行时处理输入并生成输出。

Kinesis Data Analytics 应用程序包含以下组件：

- 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：应用程序通过源 使用数据。源连接器从 Kinesis 数据流、Amazon S3 存储桶等读取数据。有关更多信息，请参阅源 (p. 14)：
- 操作符：应用程序使用一个或多个操作符 以处理数据。操作符可以转换、丰富或聚合数据。有关更多信息，请参阅DataStream API 操作员 (p. 18)：
- 接收器：应用程序使用接收器 将生成的数据发送到外部源。Sink 连接器将数据写入 Kinesis Data Streams Data Streams Data Streams Data Streams 存储Amazon S3 等。有关更多信息，请参阅接收器 (p. 15)：

在创建、、、然后，您可以创建 Kinesis Data Analytics 应用程序。您传入代码包位置、作为流数据源的 Kinesis 数据流，通常传入接收应用程序处理过的数据流或文件位置。

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，请转到步骤 1：设置 Amazon 账户并创建管理员用户 (p. 88)。

步骤 1：设置 Amazon 账户并创建管理员用户

注册一个 Amazon Web Services 账户

如果您还没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

注册 Amazon Web Services 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。

   当您注册 Amazon Web Services 账户 时，系统将会创建一个 Amazon Web Services 账户根用户。根用户有权访问该账户中的所有 Amazon Web Services和资源。作为安全最佳实践，请 为管理用户分配管理访问权限，并且只使用根用户执行 需要根用户访问权限的任务。

Amazon注册过程完成后，会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择 My Account (我的账户) 来查看当前的账户活动并管理您的账户。

保护 IAM 用户

注册 Amazon Web Services 账户 后，启用多重身份验证（MFA）保护您的管理用户。有关说明，请参阅 IAM 用户指南中的为 为 IAM 用户（控制台）启用虚拟 MFA 设备。

要授予其他用户访问您的 Amazon Web Services 账户 资源的权限，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA 并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅 IAM 用户指南中的以下主题：

- 在您的 Amazon Web Services 账户 中创建 IAM 用户
- 适用于 Amazon 资源的访问管理
- IAM 基于身份的策略示例

授权以编程方式访问

如果用户需要在 Amazon Web Services Management Console 之外与 Amazon 交互，则需要编程式访问权限。Amazon API 和 Amazon Command Line Interface 需要访问密钥。可能的话，创建临时凭证，该凭证由一个访问密钥 ID、一个秘密访问密钥和一个指示凭证何时到期的安全令牌组成。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
IAM	使用短期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中将临时凭证用于 Amazon 资源中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中管理 IAM 用户的访问密钥中的说明进行操作。

下一个步骤

步骤 2：设置 Amazon Command Line Interface (Amazon CLI) (p. 343)

步骤 2：设置 Amazon Command Line Interface (Amazon CLI)

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

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《Amazon Command Line Interface用户指南》Amazon Command Line Interface中的安装。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。有关命名配置文件的更多信息，请参阅Amazon Command Line Interface用户指南中的命名配置文件。

   ```
   [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Amazon区域的列表，请参阅《Amazon Web Services 一般参考》中的区域和终端节点。

Note

本教程中的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
```

在您设置Amazon帐户后Amazon CLI，您可以尝试下一个练习，在其中配置 end-to-end

下一个步骤

步骤：：：：：：：：：：：：：：：：：：：：：：：：：：：：：：：：：：：：：(p. 344)

步骤：：：：：：：：：：：：：：：：：：：：：：：：：：：：

在本练习中，您将创建一个以数据流作为源和汇点的 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 应用程序之前，请创建两个 Kinesis 数据流（`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Amazon CLI命令创建这些直播。有关控制台的说明，请参阅 Amazon Kinesis 数据流开发者指南中的创建和更新数据流。

创建数据流 (Amazon CLI)

1. 要创建第一个直播 (`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 应用程序代码可从中获得 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：

- 项目对象模型 (pom.xml) 文件包含有关应用程序配置和依赖关系的信息，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 源从源码流中读取数据。以下片段创建了 Kinesis 源代码：

```
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. 25)。

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. 87)。

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。确保您的项目的 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，您将在Amazon Service SSimple Storage Service (Amazon S3) ervice 存储桶中上载应用程序代码。

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. **ka-app-code-*<username>***在存储段名称字段中输入。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. 选择创建桶。
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。选择下一步。
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已存储在 Amazon S3 存储桶中，应用程序可以在其中访问它。

创建并运行 Kinesis Data Analytics 应用程序

您可以Kinesis Data Analytics 台Amazon CLI或

> Note
>
> 当您使用控制台创建应用程序时，会为您创建Amazon Identity and Access Management (IAM) 和 A CloudWatch mazon Logs 资源。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，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。
   - 将版本下拉列表保留为 Apache Flink 版本 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 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",
                "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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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：

组 ID	键	值
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**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. 要进行CloudWatch 记录，请选中 “启用” 复选框。
8. 选择更新。

> **Note**
>
> 当您选择启用亚马逊 CloudWatch 日志记录时，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 仪表板并选择所需的 Flink 作业来查看 Flink 任务图。

停止应用程序

在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，也可以从 Amazon S3 存储桶重新加载应用程序 JAR。

在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)

在本节中，您将使用创建和运行 Kinesis Data Analytics 应用程序。适用于 Apache Flink 的 Kinesis Data Analytics 使用该`kinesisanalyticsv2`Amazon CLI命令创建 Kinesis Data Analytics 应用程序并与之交互。Amazon CLI

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（您在下一节中创建）。因此，当 Kinesis Data Analytics 担任该角色时，该服务具有从源流读取和写入接收流的必要权限。

使用以下代码创建 `KAReadSourceStreamWriteSinkStream` 权限策略。*`username`*替换为用于创建 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"
        }
    ]
}
```

有关创建权限策略的 step-by-step 说明，请参阅 IAM 用户指南中的教程：创建并附加您的第一个客户托管策略。

Note

要访问其他亚马逊服务，您可以使用Amazon SDK for Java。Kinesis Data Analytics 会自动将软件开发工具包所需的证书设置为与您的应用程序关联的服务执行 IAM 角色的证书。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 应用程序可以假定该角色来读取源流并写入接收流。

未经许可，Kinesis Data Analytics 无法访问您的直播。您可以通过 IAM 角色授予这些权限。每个 IAM 角色都附加了两个策略。信任策略允许 Kinesis Data Analytics 代入角色，权限策略决定了 Kinesis Data Analytics 代入角色的权限。

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 “选择可信身份类型” 下，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 Attach permissions policies 页面上，选择 Next: Review。在创建角色后，您可以附加权限策略。
5. 在创建角色页面上**`KA-stream-rw-role`**，输入角色名称。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 Analytics 扮演这个角色，既可以从 Kinesis 数据流（源）读取数据，也可以将输出写入另一个 Kinesis 数据流。因此，您附加在上一步（the section called “创建权限策略” (p. 350)）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。

有关创建角色的 step-by-step 说明，请参阅 IAM 用户指南中的创建 IAM 角色（控制台）。

创建Kinesis Data Analytics

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 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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 日志选项

您可以使用Amazon CLI向您的应用程序添加 Amazon CloudWatch 日志流。有关在应用程序中使用 CloudWatch Logs 的信息，请参阅the section called “设置日志记录” (p. 269)。

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。在此示例中，您希望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

要加载具有相同文件名的新版本的应用程序代码，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。

要使用Amazon CLI，请从 Amazon S3 存储桶中删除您之前的代码包，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. 344)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

：：：：：：：：：：：Amazon，(p. 354)

：：：：：：：：：：：Amazon，

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：

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。

“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页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择操作，选择删除，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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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

步骤 5：后续步骤 (p. 355)

步骤 5：后续步骤

现在，您已经创建并运行了基本的 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。

- Amazon Kinesis 的Amazon流媒体数据解决方案：Amazon Kinesis 的流媒体数据解决方案可自动配置必要的Amazon服务，以便轻松捕获、存储、处理和传输流数据。Amazon该解决方案为解决流数据用例提供了多个选项。Kinesis Data Analytics 选项提供了一个 end-to-end流式传输 ETL 示例，演示了对模拟的纽约出租车数据进行分析操作的真实应用程序。该解决方案设置了所有必要的Amazon资源，例如 IAM 角色和策略、 CloudWatch 仪表板和 CloudWatch 警报。
- AmazonAmazon MSK 的Amazon流媒体数据解决方案：Amazon MSK 的流媒体数据解决方案提供了数据流经生产者、流媒体存储、消费者和目的地的Amazon CloudFormation模板。
- 带有 Apache Flink 和 Apache Kafka 的 Clickst ream Lab：一个端到端的点击流实验室，使用适用于 AManaged Streaming for Apache Kafka 的 Amazon Managed Streaming 进行流存储，使用适用于 Apache Flink 的Amazon Kinesis Data Analytics 应用程序进行流处理。
- 流媒体分析研讨会：在本研讨会中，您将构建一个 end-to-end 流媒体架构，以近乎实时地提取、分析和可视化流媒体数据。你着手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队的遥测数据，以优化其车队运营。
- 适用于 Apach Flink (p. 146)：本开发者指南的这一部分提供了在 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，可帮助您创建 Kinesis Data Analytics 应用程序和测试结果。
- 学习 Flink：动手训练：Apache Flink 官方入门培训，让你开始编写可扩展的流媒体 ETL、分析和事件驱动应用程序。

  Note

  请注意，Kinesis Data Analytics 不支持本次培训中使用的 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：包含各种 Apache Flink 应用程序示例的 GitHub 存储库。

入门指南：Flink 1.8.2

本主题包含使用 Apache Flink 1.8.2 的入门指南 (DataStream API) (p. 87)教程版本。

主题

适用于 Flink 应用程序的 Kinesis Data Analytics 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 运行时处理输入并生成输出。

Kinesis Data Analytics 应用程序包含以下组件：

- 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：应用程序通过源 使用数据。源连接器从 Kinesis 数据流、Amazon S3 存储桶等读取数据。有关更多信息，请参阅源 (p. 14)：
- 操作符：应用程序使用一个或多个操作符 以处理数据。操作符可以转换、丰富或聚合数据。有关更多信息，请参阅DataStream API 操作员 (p. 18)：
- 接收器：应用程序使用接收器 将生成的数据发送到外部源。Sink 连接器向 Kinesis Data Streame、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. 15)：

创建、编译和打包应用程序代码后，您将代码包上载到 Amazon Simple Storage Service (Amazon S3) 存储桶。然后，您创建 Kinesis Data Analytics。您传入代码包位置、作为流数据源的 Kinesis 数据流，通常传入接收应用程序处理过的数据流或文件位置。

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，你必须下载并安装 Apache Flink。有关详细信息，请参阅在之前的 Apache Flink Kinesis Streams 连接器中使用 Apache Flink (p. 323)。
- 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，请转到步骤 1：设置 Amazon 账户并创建管理员用户 (p. 357)。

步骤 1：设置 Amazon 账户并创建管理员用户

注册一个 Amazon Web Services 账户

如果您还没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

注册 Amazon Web Services 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。

   当您注册 Amazon Web Services 账户 时，系统将会创建一个 Amazon Web Services 账户根用户。根用户有权访问该账户中的所有 Amazon Web Services和资源。作为安全最佳实践，请 为管理用户分配管理访问权限，并且只使用根用户执行 需要根用户访问权限的任务。

Amazon注册过程完成后，会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择 My Account (我的账户) 来查看当前的账户活动并管理您的账户。

保护 IAM 用户

注册 Amazon Web Services 账户 后，启用多重身份验证（MFA）保护您的管理用户。有关说明，请参阅 IAM 用户指南中的为 为 IAM 用户（控制台）启用虚拟 MFA 设备。

要授予其他用户访问您的 Amazon Web Services 账户 资源的权限，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA 并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅 IAM 用户指南中的以下主题：

- 在您的 Amazon Web Services 账户 中创建 IAM 用户
- 适用于 Amazon 资源的访问管理
- IAM 基于身份的策略示例

授权以编程方式访问

如果用户需要在 Amazon Web Services Management Console 之外与 Amazon 交互，则需要编程式访问权限。Amazon API 和 Amazon Command Line Interface 需要访问密钥。可能的话，创建临时凭证，该凭证由一个访问密钥 ID、一个秘密访问密钥和一个指示凭证何时到期的安全令牌组成。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
IAM	使用短期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中将临时凭证用于 Amazon 资源中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中管理 IAM 用户的访问密钥中的说明进行操作。

步骤 2：设置 Amazon Command Line Interface (Amazon CLI)

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

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《Amazon Command Line Interface用户指南》Amazon Command Line Interface中的安装。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。有关命名配置文件的更多信息，请参阅Amazon Command Line Interface用户指南中的命名配置文件。

```
[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《Amazon Web Services e 一般参考》中的区域和终端节点。

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
```

在您设置Amazon帐户后Amazon CLI，您可以尝试下一个练习，在其中配置示例应用程序并测试 end-to-end 设置。

下一个步骤

步骤 3：为 Apache Flink 应用程序创建并运行 Kinesis Data Analytics (p. 359)

步骤 3：为 Apache Flink 应用程序创建并运行 Kinesis Data Analytics

在本练习中，您将创建一个以数据流作为源和汇点的 Kinesis Data Analytics 应用程序。

本节包含以下步骤：

- 创建两个 Amazon Kinesis Data Streams (p. 359)
- 将示例记录写入输入流 (p. 360)
- 下载并检查 Apache Flink 流式处理 Java 代码 (p. 360)
- 编译应用程序代码 (p. 361)
- 上传 Apache Flink 流式处理 Java 代码 (p. 362)
- 创建并运行 Kinesis Data Analytics 应用程序 (p. 362)
- 下一个步骤 (p. 369)

创建两个 Amazon Kinesis Data Streams

在为本练习创建 Kinesis Data Analytics 应用程序之前，请创建两个 Kinesis 数据流（`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Amazon CLI命令创建这些直播。有关控制台的说明，请参阅Amazon Kinesis 数据流开发者指南中的创建和更新数据流。

创建数据流 (Amazon CLI)

1. 要创建第一个直播 (`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 应用程序代码可从中获得 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_8目录。

请注意有关应用程序代码的以下信息：

- 项目对象模型 (pom.xml) 文件包含有关应用程序配置和依赖关系的信息，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 源从源码流中读取数据。以下片段创建了 Kinesis 源代码：

```
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. 25)。

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. 357)。

> Note
>
> 要在 1.11 之前的 Apache Flink 版本中使用 Kinesis 连接器，你需要下载、构建和安装 Apache Maven。有关更多信息，请参阅。the section called “在之前的 Apache Flink Kinesis Streams 连接器中使用 Apache Flink” (p. 323)

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

在本部分中，您创建Amazon Simple Storage Service (Amazon S3) 存储桶并上传您的应用程序代码。

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. **ka-app-code-*<username>***在存储段名称字段中输入。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. 选择创建桶。
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。选择下一步。
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已存储在 Amazon S3 存储桶中，您的应用程序可以在其中访问。

创建并运行 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，会为您创建Amazon Identity and Access Management (IAM) 和 A CloudWatch mazon Logs 资源。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. 362)
- 创建并运行应用程序 (Amazon CLI) (p. 365)

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。
   - 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 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",
                "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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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. 输入以下应用程序属性和值：

组 ID	键	值
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

5. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
6. 要进行CloudWatch 记录，请选中 “启用” 复选框。
7. 选择更新。

> **Note**
>
> 当您选择启用亚马逊 CloudWatch 日志记录时，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，也可以从 Amazon S3 存储桶重新加载应用程序 JAR。

在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)

在本节中，您将使用创建和运行 Kinesis Data Analytics 应用程序。Amazon CLI适用于 Apache Flink 的 Kinesis Data Analytics 使用该`kinesisanalyticsv2`Amazon CLI命令创建 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（您在下一节中创建）。因此，当 Kinesis Data Analytics 担任该角色时，该服务具有从源流读取和写入接收流的必要权限。

使用以下代码创建 `KAReadSourceStreamWriteSinkStream` 权限策略。*`username`*替换为用于创建 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"
        }
    ]
}
```

有关创建权限策略的 step-by-step 说明，请参阅 IAM 用户指南中的教程：创建并附加您的第一个客户托管策略。

Note

要访问其他亚马逊服务，您可以使用Amazon SDK for Java。Kinesis Data Analytics 会自动将软件开发工具包所需的证书设置为与您的应用程序关联的服务执行 IAM 角色的证书。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 应用程序可以假定该角色来读取源流并写入接收流。

未经许可，Kinesis Data Analytics 无法访问您的直播。您可以通过 IAM 角色授予这些权限。每个 IAM 角色都附加了两个策略。信任策略允许 Kinesis Data Analytics 代入角色，权限策略决定了 Kinesis Data Analytics 在代入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 “选择可信身份类型” 下，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 Attach permissions policies 页面上，选择 Next: Review。在创建角色后，您可以附加权限策略。
5. 在创建角色页面上**`KA-stream-rw-role`**，输入角色名称。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 Analytics 扮演这个角色，既可以从 Kinesis 数据流（源）读取数据，也可以将输出写入另一个 Kinesis 数据流。因此，您附加在上一步（the section called “创建权限策略” (p. 365)）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。

有关创建角色的 step-by-step 说明，请参阅 IAM 用户指南中的创建 IAM 角色（控制台）。

创建 Kinesis Data Analytics

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 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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 日志选项

您可以使用Amazon CLI向您的应用程序添加 Amazon CloudWatch 日志流。有关在应用程序中使用 CloudWatch Logs 的信息，请参阅the section called “设置日志记录” (p. 269)。

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。在此示例中，您更改了源直播和目标直播的区域。

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
>
> 要加载具有相同文件名的新版本的应用程序代码，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。

要使用Amazon CLI，请从 Amazon S3 存储桶中删除您之前的代码包，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. 359)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

步骤 4：清Amazon除 (p. 369)

步骤 4：清Amazon除

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

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，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream。
3. 在ExampleInputStream页面中，选择 “删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择 “操作”，选择 “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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。

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。

入门：Flink 1.6.2

本主题包含使用 Apache Flink 1.6.2 的入门指南 (DataStream API) (p. 87)教程版本。

主题

Amazon Kinesis Data Analyts

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 运行时处理输入并生成输出。

Amazon Kinesis Data Analytics

- 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：应用程序通过源 使用数据。源连接器从 Kinesis 数据流、Amazon S3 存储桶等读取数据。有关更多信息，请参阅源 (p. 14)：
- 操作符：应用程序使用一个或多个操作符 以处理数据。操作符可以转换、丰富或聚合数据。有关更多信息，请参阅DataStream API 操作员 (p. 18)：
- 接收器：应用程序使用接收器 将生成的数据发送到外部源。Kinesis k 连接器将Data a a a a a a a FirehKinesis Data Firehose、有关更多信息，请参阅接收器 (p. 15)：

在创建、编和打应用程序后，将代码包上传到 Amazon Smple Storage Service (Amazon S3) 存储桶。然后 Kinesis Data Analytics 建 您传入代码包位置、作为流数据源的 Kinesis 数据流，通常传入接收应用程序处理过的数据流或文件位置。

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，请转到步骤 1：设置 Amazon 账户并创建管理员用户 (p. 372)。

步骤 1：设置 Amazon 账户并创建管理员用户

注册一个 Amazon Web Services 账户

如果您还没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

注册 Amazon Web Services 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。

   当您注册 Amazon Web Services 账户 时，系统将会创建一个 Amazon Web Services 账户根用户。根用户有权访问该账户中的所有 Amazon Web Services和资源。作为安全最佳实践，请 为管理用户分配管理访问权限，并且只使用根用户执行 需要根用户访问权限的任务。

Amazon注册过程完成后，会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择 My Account (我的账户) 来查看当前的账户活动并管理您的账户。

保护 IAM 用户

注册 Amazon Web Services 账户 后，启用多重身份验证（MFA）保护您的管理用户。有关说明，请参阅 IAM 用户指南中的为 为 IAM 用户（控制台）启用虚拟 MFA 设备。

要授予其他用户访问您的 Amazon Web Services 账户 资源的权限，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA 并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅 IAM 用户指南中的以下主题：

- 在您的 Amazon Web Services 账户 中创建 IAM 用户
- 适用于 Amazon 资源的访问管理
- IAM 基于身份的策略示例

授权以编程方式访问

如果用户需要在 Amazon Web Services Management Console 之外与 Amazon 交互，则需要编程式访问权限。Amazon API 和 Amazon Command Line Interface 需要访问密钥。可能的话，创建临时凭证，该凭证由一个访问密钥 ID、一个秘密访问密钥和一个指示凭证何时到期的安全令牌组成。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
IAM	使用短期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中将临时凭证用于 Amazon 资源中的说明进行操作。
IAM	（不推荐使用）	按照《IAM 用户指南》中管理 IAM 用户的访问密钥中的说明进行操作。

哪个用户需要编程式访问权限？	目的	方式
	使用长期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	

步骤 2：设置 Amazon Command Line Interface (Amazon CLI)

在此步骤中，您可以下载并配置，Amazon CLI以与 Flink 的 Kiness Data Analytics for 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《Amazon Command Line Interface用户指南》Amazon Command Line Interface中的 “安装”。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。有关命名配置文件的更多信息，请参阅Amazon Command Line Interface用户指南中的命名配置文件。

   ```
   [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Amazon区域的列表，请参阅 Amazon Web Services 科技一般引用中的区域和终端节点。

   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
   ```

在您设置了一个Amazon帐户后Amazon CLI，您可以尝试下一个练习，在其中配置示例应用程序并测试 end-to-end 设置。

下一个步骤

步骤 3：创建并运行 Kinesis Data Analytics 应用程序 (p. 374)

步骤 3：创建并运行 Kinesis Data Analytics 应用程序

在本练习中，您将创建一个以数据流作为源和汇点的 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 应用程序之前，请创建两个 Kinesis 数据流（`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Amazon CLI命令创建这些直播。有关控制台的说明，请参阅 Amazon Kinesis 数据流开发者指南中的创建和更新数据流。

创建数据流 (Amazon CLI)

1. 要创建第一个直播 (`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_1_6目录。

请注意有关应用程序代码的以下信息：

- 项目对象模型 (pom.xml) 文件包含有关应用程序配置和依赖关系的信息，包括适用于 Apache Flink 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 源从源码流中读取数据。以下片段创建了 Kinesis 源代码：

```
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. 25)。

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. 371)。

Note

要在 1.11 之前的 Apache Flink 版本中使用 Kinesis 连接器，你需要下载连接器的源代码并按照 Apache Flink 文档中的描述进行构建。

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

     对于 Kinesis Data Analytics Runtime 版本 1.0.1，不需要-dflink.Version 参数；只有版本 1.1.0 及更高版本才需要该参数。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

在此部分，您创建了一个Amazon Smple Simple Storage Service (Amazon S3) 存储桶，并上传了应用程序代码。

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. **`ka-app-code-<username>`**在存储段名称字段中输入。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. 选择创建桶。
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。选择下一步。
9. 在设置权限步骤中，让设置保持原样。选择下一步。
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，您的应用程序可以在其中访问它。

创建并运行 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，会为您创建Amazon Identity and Access Management (IAM) 和 A CloudWatch mazon Logs 资源。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，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 在 Amazon Kinesis Data Analytics 控制面板上，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
     >
     > 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 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",
                "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)：
   - 对于 Amazon S3 存储桶，输入**ka-app-code-*<username>***。
   - 在 Amazon S3 对象的路径中，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. 输入以下应用程序属性和值：

组 ID	键	值
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

5. 在 Monitoring (监控) 下，确保 Monitoring metrics level (监控指标级别) 设置为 Application (应用程序)。
6. 要进行CloudWatch 记录，请选中 “启用” 复选框。
7. 选择更新。

> **Note**
>
> 当您选择启用亚马逊 CloudWatch 日志记录时，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，也可以从 Amazon S3 存储桶重新加载应用程序 JAR。

在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)

在本节中，您将使用创建和运行 Kinesis Data Analytics 应用程序。Amazon CLI适用于 Apache Flink 的 Kinesis Data Analytics 使用该`kinesisanalyticsv2`Amazon CLI命令创建 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（您在下一节中创建）。因此，当 Kinesis Data Analytics 担任该角色时，该服务具有从源流读取和写入接收流的必要权限。

使用以下代码创建 `KAReadSourceStreamWriteSinkStream` 权限策略。*`username`*替换为用于创建 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"
        }
    ]
}
```

有关创建权限策略的 step-by-step 说明，请参阅 IAM 用户指南中的教程：创建并附加您的第一个客户托管策略。

> **Note**
>
> 要访问其他亚马逊服务，您可以使用Amazon SDK for Java。Kinesis Data Analytics 会自动将软件开发工具包所需的证书设置为与您的应用程序关联的服务执行 IAM 角色的证书。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 应用程序可以假定该角色来读取源流并写入接收流。

未经许可，Kinesis Data Analytics 无法访问您的直播。您可以通过 IAM 角色授予这些权限。每个 IAM 角色都附加了两个策略。信任策略授予 Kinesis Data Analytics a Kinesis Data Analytics，权限策略决定了

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 “选择可信身份类型” 下，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 Attach permissions policies 页面上，选择 Next: Review。在创建角色后，您可以附加权限策略。
5. 在创建角色页面上**`KA-stream-rw-role`**，输入角色名称。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 Analytics 扮演这个角色，既可以从 Kinesis 数据流（源）读取数据，也可以将输出写入另一个 Kinesis 数据流。因此，您附加在上一步（the section called “创建权限策略” (p. 380)）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。

有关创建角色的 step-by-step 说明，请参阅 IAM 用户指南中的创建 IAM 角色（控制台）。

Kinesis Data Analytics

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 CloudWatch 控制台上查看 Kinesis Data Analytics 指标，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 日志选项

您可以使用Amazon CLI向您的应用程序添加 Amazon CloudWatch 日志流。有关在应用程序中使用 CloudWatch Logs 的信息，请参阅the section called “设置日志记录” (p. 269)。

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。在此示例中，您更改了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。

要使用Amazon CLI，请从 Amazon S3 存储桶中删除之前的代码包，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。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. 374)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"
                }
            }
        }
    }
}
```

步骤 4：清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：

- Kinesis Data Analytics (p. 384)
- 删除Kinesis Data Streams (p. 384)
- 删除您的Amazon S3 对象和存储桶 (p. 385)
- 删除您的 IAM 资源 (p. 385)
- 删除您的 CloudWatch 资源 (p. 385)

Kinesis Data Analytics

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，网址为 https://console.aws.amazon.com/kinesisanalytics。
2. 在 Kinesis Data Streams 面板中，选择ExampleInputStream。
3. 在ExampleInputStream页面中，选择“删除 Kinesis Stream”，然后确认删除。
4. 在 Kinesis stream s 页面中 ExampleOutputStream，选择，选择“操作”，选择“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- 存储桶。 <username>
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

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>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 k inesis-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

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 主机。
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

Kinesis Data Analytics for Apache Flink Kinesis Data Analytics 使用本节中描述的默认值。其中一些值可以由 Kinesis Data Analytics 应用程序在代码中设置，而其他值则无法更改。

本主题包含下列部分：

Apache Flink 配置

Kinesis Data Analytics 提供了默认 Flink 配置，其中包含 Apache Flink 针对大多数属性的推荐值和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程序。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、高内存和状态使用率的应用程序的性能，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 控制面板检查应用程序的当前配置。

状态后端

Kinesis Data Analytics 将临时数据存储在状态后端。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，默认情况下`state.backend.rocksdb.ttl.compaction.filter.enabled`配置处于启用状态。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 Apache Flink 文档中的 F link 1.8.0 中的状态 TTL。

有关状态后端的更多信息，请参阅 Apache Flink 文档中的状态后端。

检查点

Data Analytics for Apache Flink 可以更改其中的一些值。你必须设置CheckpointConfiguration。ConfigurationType改`CUSTOM`为让 Kinesis Data Analytics 使用修改后的检查点值。

设置	是否可以修改？	默认值
CheckpointingEnabled	可修改	True
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
未对齐的检查点	不能修改	0

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。您可以通过将应用程序的AllowNonRestoredState参数设置为来FlinkRunConfiguration允许操作成功`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 Apache Flink 文档中的 Allowing Non-Restored State（允许未还原状态）。

堆大小

Kinesis Data Analytics 为每个 KPU 分配 3 GiB 的 JVM 堆，并预留 1 GiB 用于原生代码分配。有关增加应用程序容量的信息，请参阅the section called “扩缩” (p. 33)。

有关 JVM 堆大小的更多信息，请参阅 A pache Flink 文档中的配置。

可修改的 Flink 配置属性

以下是您可以使用支持案例修改的 Flink 配置设置。您可以一次修改多个属性，也可以通过指定应用程序前缀来同时修改多个应用程序的属性。如果您想修改此列表之外的其他 Flink 配置属性，请指定您的具体属性。

容错能力

```
restart-strategy:
```

restart-strategy.fixed-delay.delay:

检查点和状态后端

state.backend:

state.backend.fs.memory-threshold:

state.backend.incremental:

rockesit 原生指标

state.backend.rocksdb.compaction.style:

state.backend.rocksdb.memory.partitioned-index-filters:

state.backend.rocksdb.metrics.actual-delayed-write-rate:

state.backend.rocksdb.metrics.background-errors:

state.backend.rocksdb.metrics.block-cache-capacity:

state.backend.rocksdb.metrics.block-cache-pinned-usage:

state.backend.rocksdb.metrics.block-cache-usage:

state.backend.rocksdb.metrics.column-family-as-variable:

state.backend.rocksdb.metrics.compaction-pending:

state.backend.rocksdb.metrics.cur-size-active-mem-table:

state.backend.rocksdb.metrics.cur-size-all-mem-tables:

state.backend.rocksdb.metrics.estimate-live-data-size:

state.backend.rocksdb.metrics.estimate-num-keys:

state.backend.rocksdb.metrics.estimate-pending-compaction-bytes:

state.backend.rocksdb.metrics.estimate-table-readers-mem:

state.backend.rocksdb.metrics.is-write-stopped:

state.backend.rocksdb.metrics.mem-table-flush-pending:

state.backend.rocksdb.metrics.num-deletes-active-mem-table:

state.backend.rocksdb.metrics.num-deletes-imm-mem-tables:

state.backend.rocksdb.metrics.num-entries-active-mem-table:

state.backend.rocksdb.metrics.num-entries-imm-mem-tables:

state.backend.rocksdb.metrics.num-immutable-mem-table:

state.backend.rocksdb.metrics.num-live-versions:

state.backend.rocksdb.metrics.num-running-compactions:

state.backend.rocksdb.metrics.num-running-flushes:

state.backend.rocksdb.metrics.num-snapshots:

state.backend.rocksdb.metrics.size-all-mem-tables:

state.backend.rocksdb.thread.num:

高级状态后端选项

state.storage.fs.memory-threshold:

完整 TaskManager 选项

task.cancellation.timeout:

taskmanager.jvm-exit-on-oom:

taskmanager.numberOfTaskSlots:

taskmanager.slot.timeout:

taskmanager.network.memory.fraction:

taskmanager.network.memory.max:

taskmanager.network.request-backoff.initial:

taskmanager.network.request-backoff.max:

内存配置

taskmanager.heap.size:

taskmanager.memory.jvm-metaspace.size:

taskmanager.memory.jvm-overhead.fraction:

taskmanager.memory.jvm-overhead.max:

taskmanager.memory.managed.consumer-weights:

taskmanager.memory.managed.fraction:

taskmanager.memory.network.fraction:

taskmanager.memory.network.max:

taskmanager.memory.segment-size:

taskmanager.memory.task.off-heap.size:

RPC /Akka

akka.ask.timeout:

akka.client.timeout:

akka.framesize:

akka.lookup.timeout:

akka.tcp.timeout:

客户端

client.timeout:

高级集群选项

cluster.intercept-user-system-exit:

cluster.processes.halt-on-fatal-error:

文件系统配置

fs.s3.connection.maximum:

fs.s3a.connection.maximum:

fs.s3a.threads.max:

s3.upload.max.concurrent.uploads:

高级容错选项

heartbeat.timeout:

jobmanager.execution.failover-strategy:

内存配置

jobmanager.memory.heap.size:

指标

metrics.latency.interval:

REST 端点和客户端的高级选项

rest.flamegraph.enabled:

rest.server.numThreads:

高级 SSL 安全选项

security.ssl.internal.handshake-timeout:

高级日程安排选项

slot.request.timeout:

Flink Web 界面的高级选项

`web.timeout:`

查看配置的 Flink 属性

你可以通过 Apache Flink 控制面板查看你自己配置的或通过支持案例请求修改的 Apache Flink 属性，然后按照以下步骤操作：

1. 前往 Flink 控制面板
2. 在左侧导航窗格中选择 Job 管理器。
3. 选择 “配置” 查看 Flink 属性列表。

为 Aa Link 配置 Kinesis Da Analytics

您可以配置 Kinesis Da Analytics，使其连接到您账户中虚拟私有云（VPC）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：

AVPC 念念念念念

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：

- 虚拟私有云 (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。
- VPC 终端节点使您能够将 VPC 私密地连接到支持的Amazon服务和 VPC 终端节点服务（由提供支持），PrivateLink 而无需使用互联网网关、NAT 设备、VPNAmazon Direct Connect 连接或连接。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 AVPC 服务的更多信息，请参阅 AAmazon V irtual Private Cloud 服务和。

Kinesis Data Analytics 在应用程序的 VPC 配置中提供的其中一个子网中创建弹性网络接口。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. 33)。

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 Amazon Kinesis Data Analytics (p. 254)。

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。

访问亚马逊 VPC 的权限政策

```
{
  "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

当您使用控制台（例如 CloudWatch 日志或 Amazon VPC）指定应用程序资源时，控制台会修改您的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。

连接到 VPC 的 Kinesis Data Analytics 应用程序的互联网和服务访问权限

默认情况下，在将 Kinesis Da Analytics 连接到您账户中的 VPC 时，除非 VPC 提供访问权限，否则它无需 Internes 访问权限。如果应用程序需要访问 Internet，则需要满足以下条件：

- Kinesis Data Analytics 应用程序只能配置为私有子网。
- 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As 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：目的地：0.0.0.0/0，目标：igw-…
  - 私有子网：目的地：0.0.0.0/0，目标：nat-… 或 eni-…

要将 Kinesis Data Analytics 应用程序与私有子网关联，请执行以下操作：

- 打开 Kinesis Data Analytics 控制台，网址为 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

使用以下 Kinesis Data Analytics API 操作来管理应用程序的 VPC。有关使用 Kinesis Data Analytics API 的信息，请参阅API 示例代码 (p. 423)。

CreateApplication

在创建过程中，使用CreateApplication操作将 VPC 配置添加到您的应用程序。

`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_15",
  "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

创建 VPC 配置后，使用AddApplicationVpcConfiguration操作将其添加到您的应用程序。

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操作从您的应用程序中删除 VPC 配置。

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操作一次性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
}
```

示例：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. 167)。

排查在AKinesis Data Analytics

以下可帮助您排查在Azzon Kinesis Datata Analyyyyyicics AAzon Kinesis Analyyyicics A

主题

开发故障排除

主题

ApachachachchFlink

默认情况下，在支持 Flame Graphs 的 Apache Flink 版本的 Kinesis Data Analytics 应用程序上启用火焰图表。如 Flink文档中所述，如果你保持图表处于打开状态，F lame Graps可能会影响应用程序的性能。

如果您想为应用程序禁用 Flame Graphs，请创建一个案例以请求为您的应用程序 ARN 禁用它。有关更多信息，请参阅Supp Amazonort 中心。

EFO 连接器 1.15.2 中的凭证提供者问题

1.15.2 之前的 Kinesis Data Streams EFO 连接器版本存在一个已知问题，`FlinkKinesisConsumer`即不尊重`Credential Provider`配置。由于该问题，有效配置被忽略，这会导致使用AUTO凭证提供商。这可能会导致使用 EFO 连接器跨账户访问 Kinesis 时出现问题。

要解决此错误，请使用 EFO 连接器版本 1.15.3 或更高版本。

带有不支持 Kinesis 连接器的应用程序

如果应用程序使用捆绑在应用程序 JAR 或存档 (ZIP) 中的不受支持的 Kinesis Connector 版本（1.15.2 之前的版本），则 Kinesis Data Analytics for Apache Flink 版本 1.15 将自动拒绝应用程序启动或更新。

拒绝错误

通过以下方式提交创建/更新应用程序调用时，您将看到以下错误：

```
An error occurred (InvalidArgumentException) when calling the CreateApplication operation:
 An unsupported Kinesis connector version has been detected in the application. Please
 update flink-connector-kinesis to any version equal to or newer than 1.15.2.
For more information refer to connector fix: https://issues.apache.org/jira/browse/
FLINK-23528
```

补救步骤

- 更新应用程序对的依赖关系`flink-connector-kinesis`。如果您使用 Maven 作为项目的构建工具，请关注更新 Maven 依赖关系 (p. 398)。如果你使用的是 Gradle，请关注更新 Gradle 依赖关系 (p. 399)。
- 重新打包应用程序。
- 将 Go 更新为Amazon S3 存储桶。
- 使用刚刚上传到 Amazon S3 存储桶的修改后的应用程序重新提交创建/更新应用程序请求。
- 如果您继续看到相同的错误消息，请重新检查您的应用程序依赖关系。如果问题仍然存在，请创建支持请求。

更新 Maven 依赖关系

1. 打开项目的`pom.xml`。
2. 查找项目的依赖关系。它们看起来像：

```
<project>

    ...

    <dependencies>

        ...

        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-connector-kinesis</artifactId>
        </dependency>

        ...

    </dependencies>

    ...

</project>
```

3. 更新`flink-connector-kinesis`到等于或高于 1.15.2 的版本。例如：

```
<project>

    ...

    <dependencies>

        ...

        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-connector-kinesis</artifactId>
            <version>1.15.2</version>
        </dependency>

        ...

    </dependencies>

    ...
```

```
</project>
```

更新 Gradle 依赖关系

1. 打开项目的build.gradle（或build.gradle.kts适用于 Kotlin 应用程序）。
2. 查找项目的依赖关系。它们看起来像：

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis")

    ...

}

...
```

3. 更新flink-connector-kinesis到等于或高于 1.15.2 的版本。例如：

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis:1.15.2")

    ...

}

...
```

编译错误：“无法解析项目的依赖关系”

要编译适用于 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 “在之前的 Apache Flink Kinesis Streams 连接器中使用 Apache Flink” (p. 323)。

无效的选择：“kinesisanalyticsv2”

要使用 Kinesis Data Analytics API 的 v2，你需要最新版本的Amazon Command Line Interface (Amazon CLI)。

有关升级的信息Amazon CLI，请参阅《Amazon Command Line Interface用户指南》Amazon Command Line Interface中的安装。

UpdateApplication 操作不是在重新加载应用程序代码

如果未指定 S3 对象版本，则该UpdateApplication操作将不会重新加载具有相同文件名的应用程序代码。要重新加载具有相同文件名的应用程序代码，请在 S3 存储桶上启用版本控制，并使用`ObjectVersionUpdate`参数指定新的对象版本。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。

运行时故障排查

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，您可以为指示应用程序中的错误或瓶颈状况的 CloudWatch 指标设置阈值。有关推荐 CloudWatch 警报的信息，请参阅在 Amazon Kinesis Data Analytics 中使用 CloudWatch 警报 (p. 290)。

应用程序问题

本节包含您在使用 Kinesis Data Analytics 应用程序时可能遇到的错误情况的解决方案。

主题

- java.nio.file。 NoSuchFileException: /usr/local/openjdk-8/lib/security/cacerts (p. 403)

应用程序停留在临时状态

如果您的应用程序保持临时状态（STARTINGUPDATING、STOPPING、或AUTOSCALING），则可以使用Force参数设置为的StopApplication操作来停止应用程序true。您无法强制停止处于该DELETING状态的应用程序。或者，如果应用程序处于UPDATING或AUTOSCALING状态，则可以将其回滚到之前的运行版本。回滚应用程序时，它会加载上次成功快照中的状态数据。如果应用程序没有快照，Kinesis Data Analytics 会拒绝回滚请求。有关回滚应用程序的更多信息，请参阅RollbackApplication操作。

> Note
>
> 强制停止应用程序可能会导致数据丢失或重复。为防止在应用程序重启期间丢失数据或重复处理数据，我们建议您经常拍摄应用程序的快照。

应用程序卡住的原因包括：

- 应用程序状态太大：应用程序状态过大或过于持久会导致应用程序在检查点或快照操作期间卡住。检查您的应用程序lastCheckpointDuration和lastCheckpointSize指标中是否存在稳步增加的值或异常高的值。
- 应用程序代码太大：验证您的应用程序 JAR 文件是否小于 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UpdateApplication或StopApplication请求期间拍摄应用程序快照。然后，该服务使用此快照状态并使用更新的应用程序配置恢复应用程序，以提供完全一次的处理语义。如果自动快照创建失败，请参见创建快照失败 (p. 401)下文。
- 从快照恢复失败：如果您在应用程序更新中删除或更改操作员并尝试从快照恢复，则默认情况下，如果快照包含缺失操作员的状态数据，则恢复将失败。此外，应用程序将停留在STOPPED或UPDATING状态。要更改此行为并允许恢复成功，请将应用程序的AllowNonRestoredState参数更改FlinkRunConfiguration为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（软设置）。如果您的任务未能在此超时时间内启动，您将看到如下 CloudWatch 日志：

```
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，这意味着你在 Flink 作业main方法下定义的操作需要超过 5 分钟，导致 Flink 任务在 Kinesis Data Analytics 端创建超时。我们建议您查看 Flink JobManager日志和应用程序代码，看看该main方法的延迟是否在预料之中。如果不是，则需要采取措施解决问题，以便在 5 分钟内完成。

您可以使用ListApplications或DescribeApplication操作检查您的申请状态。

创建快照失败

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. 30)：
- 该应用程序无权访问其源代码或接收器。
- 应用程序代码无法正常运行。
- 应用程序遇到其他配置问题。

如果您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，请将应用程序的SnapshotsEnabled属性设置为 ApplicationSnapshotConfiguration，false然后重试请求。

如果未正确配置应用程序的操作员，则快照可能会失败。有关调整操作员性能的信息，请参见操作员扩展 (p. 303)。

在应用程序恢复正常状态后，我们建议您将应用程序的SnapshotsEnabled属性设置为true。

无法访问 VPC 中的资源

如果您的应用程序使用在 Amazon VPC 上运行的 VPC，请执行以下操作以验证您的应用程序是否有权访问其资源：

- 检查您的 CloudWatch 日志中是否存在以下错误。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 CloudWatch 日志的信息，请参阅日志记录和监控 (p. 268)。

写入 Amazon S3 3 时数据丢失

使用 Apache Flink 版本 1.6.2 将输出写入Amazon S3 存储桶时，可能会丢失一些数据。我们建议在使用 Amazon S3 直接输出时使用 Apache Flink 支持的最新版本。要使用 Apache Flink 1.6.2 写入Amazon S3 存储桶，我们建议使用 Kinesis Data Firehose。有关将 Kinesis Data Firehose 与 Kinesis Data Analytics 相关的更多信息，请参阅Kinesis Data Firehose (p. 178)。

应用程序处于 RUNNING 状态但未处理数据

您可以使用ListApplications或DescribeApplication操作来检查您的申请状态。如果您的应用程序进入RUNNING状态但未将数据写入您的接收器，则可以通过向应用程序添加亚马逊 CloudWatch 日志流来解决问题。有关更多信息，请参阅使用应用程序 CloudWatch 日志选项 (p. 271)：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，可能会出现与以下内容类似的错误：

```
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，也可以使用操作SnapshotsEnabledUpdate参数来执行此UpdateApplication操作。
- 调查无法创建快照的原因。有关更多信息，请参阅应用程序停留在临时状态 (p. 401)：
- 当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

SSL 信任库的位置已在先前的部署中更新。请在 `ssl.truststore.location` 参数中改用以下值：

```
/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
```

应用程序正在重新启动

如果你的应用程序运行不正常，它的 Apache Flink 作业会不断失败并重新启动。本节介绍这种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可能为以下症状：

- 该`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。
- 该`Downtime`指标不是零。此指标表示应用程序处于`FAILING`或`RESTARTING`状态的毫秒数。
- 应用程序日志包含对`RESTARTING`或的状态更改`FAILED`。您可以使用以下Logs Insights查询查询在应用程序 CloudWatch 日志中查询这些状态更改:分析错误：应用程序的任务相关故障 (p. 277).

原因和解决方法

以下情况可能会导致您的应用程序变得不稳定并反复重启：

- 操作员抛出异常：如果您的应用程序中某个运算符中的任何异常未得到处理，则应用程序会进行故障转移（通过解释为操作员无法处理故障）。应用程序从最新的检查点重新启动，以保持 “一劳永逸” 的处理语义。因此，`Downtime`在这些重启期间不为零。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. 277)：
- Kinesis Data Streams 的配置不正确：如果应用程序的源或接收器是 Kinesis 数据流，请检查该流的指标是否为`ReadProvisionedThroughputExceeded`或`WriteProvisionedThroughputExceeded`错误。

  如果您看到这些错误，则可以通过增加流的分片数量来增加 Kinesis 流的可用吞吐量。有关更多信息，请参阅如何更改 Kinesis Data Streams 中打开的分片数量？ 。
- 其他源或接收器未正确配置或不可用：验证您的应用程序是否正确配置了源和接收器。检查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）是否配置良好，没有遇到读取或写入限制，或者是否定期不可用。

  如果您的依赖服务遇到与吞吐量相关的问题，请增加这些服务的可用资源，或者调查任何错误或不可用的原因。
- 运算符未正确配置：如果您的应用程序中某个运算符的线程工作负载未正确分配，则该运算符可能会过载，应用程序可能会崩溃。有关调整运算符并行性的信息，请参见正确管理操作员扩展 (p. 303)。
- 应用程序失败 DaemonException：如果您使用的是 1.11 之前的 Apache Flink 版本，则此错误会出现在您的应用程序日志中。你可能需要升级到 Apache Flink 的更高版本，这样才能使用 0.14 或更高版本的 KPL 版本。
- 应用程序失败 TimeoutException，显示 FlinkException、或 RemoteTransportException：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。如果您的应用程序过载，您的任务管理器可能会承受 CPU 或内存资源压力，从而导致它们失败。

这些错误可能为以下与以下相相相相相相相似：

- `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`

要解决此问题，请检查以下内容：

- 检查您的 CloudWatch 指标，看看 CPU 或内存使用率是否出现异常峰值。
- 检查您的应用程序是否存在吞吐量问题。有关更多信息，请参阅性能排查的问题 (p. 301)：
- 检查应用程序日志，查看应用程序代码引发的未处理异常。

- 应用程序失败并出现 “ JaxbAnnotationModule 未找到” 错误：如果您的应用程序使用 Apache Beam，但没有正确的依赖项或依赖项版本，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关系：

```
<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，则您的应用程序会从环境依赖项中加载该版本，并且由于版本不匹配，应用程序会在运行时崩溃。

有关将 Apache Beam Kinesis Data Analytics，请参阅CloudFormation 与 Kinesis Data Analytics (p. 136)。

吞吐量太慢

如果你的应用程序处理传入的流数据的速度不够快，它将表现不佳并变得不稳定。本节介绍这种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：

- 如果您的应用程序的数据源是 Kinesis 流，则该流的`MillisBehindLatest`指标会不断增加。
- 如果您的应用程序的数据源是 Amazon MSK 集群，则该集群的消费延迟指标会持续增加。有关更多信息，请参阅 Amazon MS3 开发人员指南中的消费者延迟监控。
- 如果您的应用程序的数据源是不同的服务或来源，请检查任何可用的消费者滞后指标或可用的数据。

原因和解决方法

应用程序吞吐量缓慢的原因可能有很多。如果您的应用程序跟不上输入的速度，请检查以下内容：

- 如果吞吐量延迟达到峰值然后逐渐减小，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，从而导致延迟加剧。有关应用程序故障的信息，请参见应用程序正在重新启动 (p. 403)。
- 如果吞吐量延迟一致，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，请参阅性能排查的问题 (p. 301)。

- 如果吞吐量延迟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，请参见扩缩 (p. 33)。

有关应用程序源中吞吐量慢或使用者延迟增加的故障排除步骤，请参阅性能排查的问题 (p. 301)。

州无界增长

如果你的应用程序没有正确处置过时的状态信息，它会不断积累并导致应用程序性能或稳定性问题。本节介绍这种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：

- 该`lastCheckpointDuration`指标正在逐渐增加或激增。
- 该`lastCheckpointSize`指标正在逐渐增加或激增。

原因和解决方法

以下情况可能会导致您的应用程序积累状态数据：

- 您的应用程序保留状态数据的时间超过了所需的时间。
- 您的应用程序使用的窗口查询持续时间过长。
- 您没有为状态数据设置 TTL。有关更多信息，请参阅 Apache Flink 文档中的状态生存时间 (TTL)。
- 你正在运行一个依赖于 Apache Beam 2.25.0 或更高版本的应用程序。您可以通过扩展关键实验和价值来选择退出读取转换的新版本`use_deprecated_read`。 BeamApplicationProperties有关更多信息，请参阅 Apache Beam 文档。

有时应用程序会面临不断增长的状态大小增长，从长远来看这是不可持续的（毕竟 Flink 应用程序可以无限期运行）。有时，这可以追溯到应用程序在状态下存储数据而没有正确老化旧信息。但有时人们对 Flink 能提供什么抱有不合理的期望。应用程序可以在跨越数天甚至数周的大时间窗口内使用聚合。AggregateFunctions除非使用允许增量聚合，否则 Flink 需要保持整个窗口的事件处于状态。

此外，在使用流程函数实现自定义运算符时，应用程序需要从状态中删除业务逻辑不再需要的数据。在这种情况下，状态 time-to-live可用于根据处理时间自动使数据老化。Kinesis Data Analytics 使用增量检查点，因此状态 ttl 基于 RocksDB 压缩。只有在压缩操作发生后，您才能观察到状态大小（由检查点大小表示）的实际减小。特别是对于小于 200 MB 的检查点大小，您不太可能因为状态到期而观察到任何检查点大小减小。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 中触发快照以强制移除过时状态。

出于调试目的，禁用增量检查点以更快地验证检查点大小是否确实减小或稳定（并避免 RockSB 中压缩的影响）是有意义的。不过，这需要向服务团队出票。

I/O 绑定操作符

最好避免在数据路径上依赖外部系统。与查询外部系统来丰富单个事件相比，将参考数据集保持在状态通常要高得多。但是，有时有些依赖关系无法轻松转移到状态，例如，如果您想使用托管在 Amazon Sagemaker 上的机器学习模型来丰富事件。

通过网络与外部系统连接的运营商可能会成为瓶颈，并造成背压。强烈建议使用 AsyncIO 来实现该功能，以减少单个调用的等待时间并避免整个应用程序变慢。

此外，对于具有 I/O 绑定运算符的应用程序，增加 Kinesis Data Analytics 应用程序的 ParallelismPerKPU 设置也是有意义的。此配置描述应用程序在其使用的每个 Kinesis 处理单元（KPU）可以执行的parallel 子任务

数。通过将值从默认值 1 增加到（比如 4），应用程序利用相同的资源（且成本相同），但可以扩展到并行度的 4 倍。这适用于绑定 I/O 的应用程序，但会给非 I/O 绑定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：应用程序遇到`LimitExceededExceptions`来自其上游源 Kinesis 数据流的情况。

潜在原因：Apache Flink 库 Kinesis 连接器的默认设置设置为从 Kinesis 数据流源读取，而每次`GetRecords`调用获取的最大记录数的默认设置非常激进。Apache Flink 默认配置为每次 GetRecords 调用 10,000 条记录（默认情况下，此调用每 200 ms 进行一次），尽管每个分片的限制只有 1,000 条记录。

在尝试从 Kinesis 数据流中消耗数据时，这种默认行为可能会导致节流，这将影响应用程序的性能和稳定性。

可以通过检查 CloudWatch `ReadProvisionedThroughputExceeded`指标并查看该指标长期或持续大于零的时间段来确认这一点。

客户还可以在其 Kinesis Analytics Flink 应用程序 CloudWatch 的日志中看到持续的`LimitExceededException`错误。

解决方案：客户可以采取以下两项措施之一来解决此情况：

- 降低每次 GetRecords 调用获取的记录数的默认限制
- 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_RE ADS

检查点

检查点是 Flink 确保应用程序状态容错的机制。该机制允许 Flink 在任务失败时恢复运算符的状态，并为应用程序提供与无故障执行相同的语义。使用 Kinesis Data Analytics，应用程序的状态存储在 RocksDB 中，这是一种在磁盘上保持其工作状态的嵌入式键/值存储。创建检查点时，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，请参阅状态快照的工作原理？。

通过检查点检验阶段

对于 Flink 中的检查点运算符子任务，有 5 个主要阶段：

- 等待 [Start Delay] — Flink 使用插入到直播中的检查点屏障，所以这个阶段的时间就是操作员等待检查点屏障到达的时间。
- 对齐 [对齐持续时间] — 在此阶段，子任务已达到一个屏障，但它正在等待来自其他输入流的障碍。
- Sync checkpoint [Sync D uration] — 此阶段是子任务实际快照操作员的状态并阻止子任务上的所有其他活动的阶段。
- 异步检查点 [Async 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，只是子任务向发送确认 JobManager 并执行任何提交消息（例如，使用 Kafka sinks）。

每个阶段（确认除外）都映射到检查点的持续时间指标，该指标可从 Flink WebUI 获得，这有助于找出长检查点的原因。

要查看检查点上每个可用指标的确切定义，请前往 “历史记录” 选项卡。

正在调查

在调查检查点持续时间较长时，需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到达检查点的时间最长，以及该子任务的哪个阶段需要更长的时间。这可以通过作业检查点任务下的 Flink WebUI 来确定。Flink 的 Web 界面提供有助于调查检查点问题的数据和信息。有关完整明细，请参阅监控检查点。

首先要看的是Job 图中每个操作员的端到端持续时间，以确定哪个操作员需要很长时间才能到达检查点，需要进一步调查。根据 Flink 文档，持续时间的定义是：

从触发时间戳到最新确认的持续时间（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最后一个确认检查点的子任务决定。这个时间通常比单个子任务实际检查状态所需的时间要长。

检查点的其他持续时间也提供了有关在哪里花费时间的更精细的信息。

如果同步持续时间很长，则表示在快照期间发生了一些事情。在此阶段调用实现 SnapshotState 接口的类；这可以`snapshotState()`是用户代码，因此线程转储可用于对此进行研究。

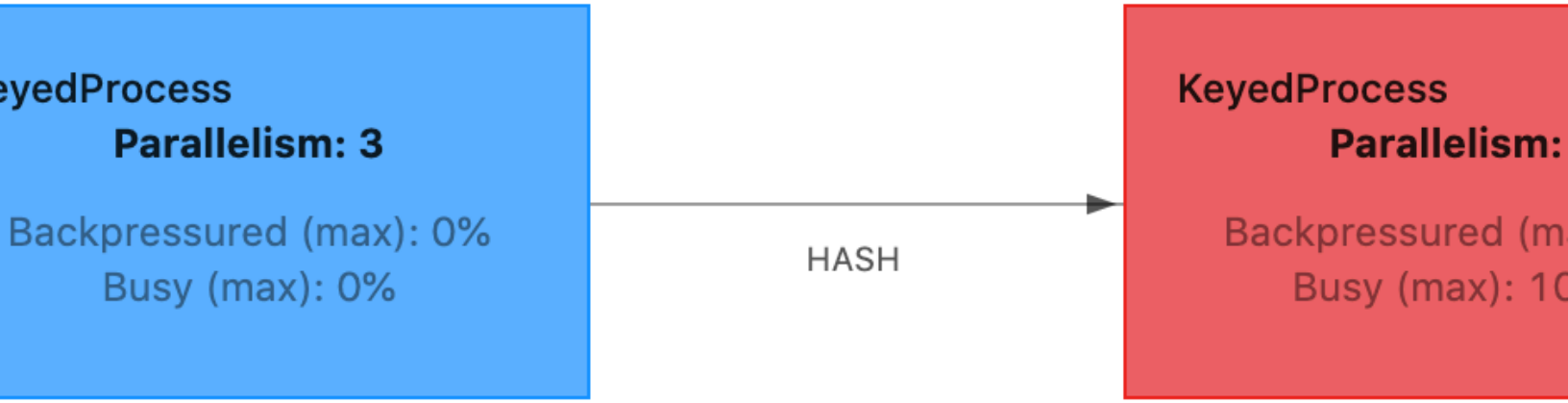
较长的异步持续时间表明需要花费大量时间将状态上传到 Amazon S3。如果状态很大，或者正在上传很多状态文件，就会发生这种情况。如果是这样的话，那么值得研究一下应用程序是如何使用状态的，并确保尽可能使用 Flink 原生数据结构（使用键控状态）。Kinesis Data Analytics 配置 Flink 的方式可以最大限度地减少 Amazon S3 的调用次数，从而确保调用时间不会太长。以下是操作员的检查点统计信息示例。它表明，与之前的运算符检查点统计数据相比，Async 持续时间相对较长。

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

如果启动延迟过高，则表明大部分时间都花在等待检查点屏障到达操作员手中。这表明应用程序需要一段时间来处理记录，这意味着任务图中的障碍正在缓慢流动。如果Job 压力过大或操作员经常忙碌，通常会出现这种情况。以下是第二位 JobGraph KeyedProcess 操作员忙碌的示例。


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 线程转储来调查花了这么长时间的原因。一旦确定了瓶颈，就可以使用 Flame-graph 或 thread-dumps 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 flame graph。线程转储输出所有线程在某个时间点的执行状态。Flink 采用 JVM 线程转储，这是 Flink 进程内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附加信息来呈现。Flame 图实际上是使用快速连续获取的多个堆栈跟踪来构建的。该图是由这些跟踪结果制成的可视化，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 中为单个线程提取的线程转储片段。第一行包含有关该话题的一些一般信息，包括：

- 话题名称 KeyedProcess (1/3) #0
- 话题的优先级 p rio=5
- 一个唯一的话题 ID =1423
- 线程状态可运行

话题的名称通常提供有关该话题一般用途的信息。操作员线程可以通过其名称来识别，因为操作员线程与操作员线程同名，还可以指示它与哪个子任务相关，例如，KeyedProcess (1/3) #0 线程来自KeyedProcess操作员并且来自第 1 个（共 3 个）子任务。

线程可能处于以下几种几种几种某种状态：

- 新 — 话题已创建但尚未处理
- 可运行 — 线程在 CPU 上执行
- BLOCKED — 该线程正在等待另一个线程释放其锁定
- 等待-线程正在使用`wait()`、`join()`、或`park()`方法等待
- TIMED_WAITING — 线程使用休眠、等待、加入或暂留方法等待，但等待时间最长。

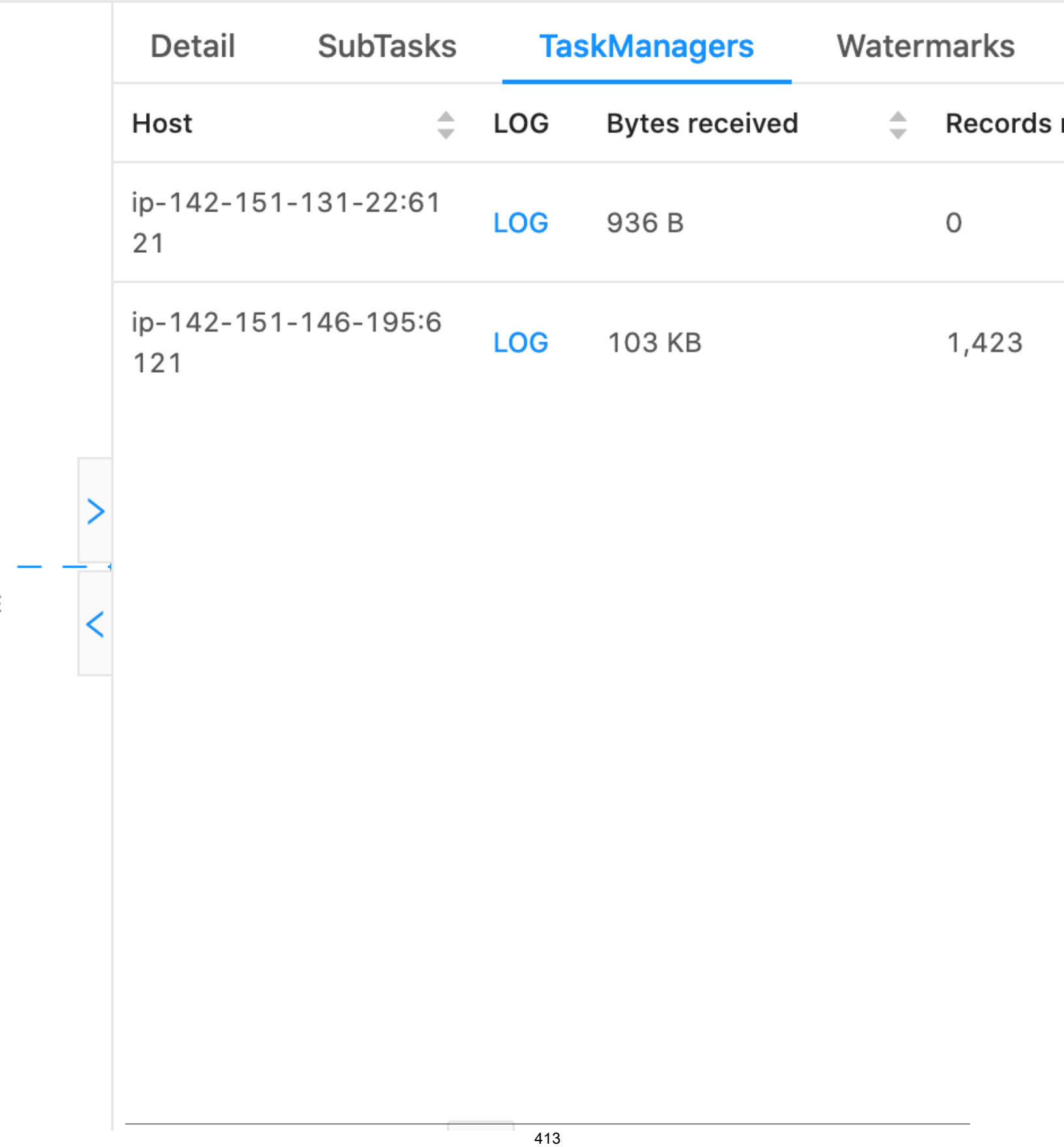
> **Note**
>
> 在 Flink 1.13 中，线程转储中单个堆栈跟踪的最大深度限制为 8。

> **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，因为它们可能难以阅读，需要采集多个样本和手动分析。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 Flink 界面左侧导航栏上的任务管理器选项，选择特定的任务管理器，然后导航到 Thread D ump 选项卡来进行线程转储。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），也可以直接在 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。

选择特定的运算符时，可以使用哪个任务管理器对TaskManagers选项卡进行线程转储。这表明操作员正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，与操作员有关的垃圾最为重要。这些很容易找到，因为运算符线程与运算符同名，并且表明它与哪个子任务有关。例如，以下堆栈跟踪来自KeyedProcess操作员，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
    ...
```

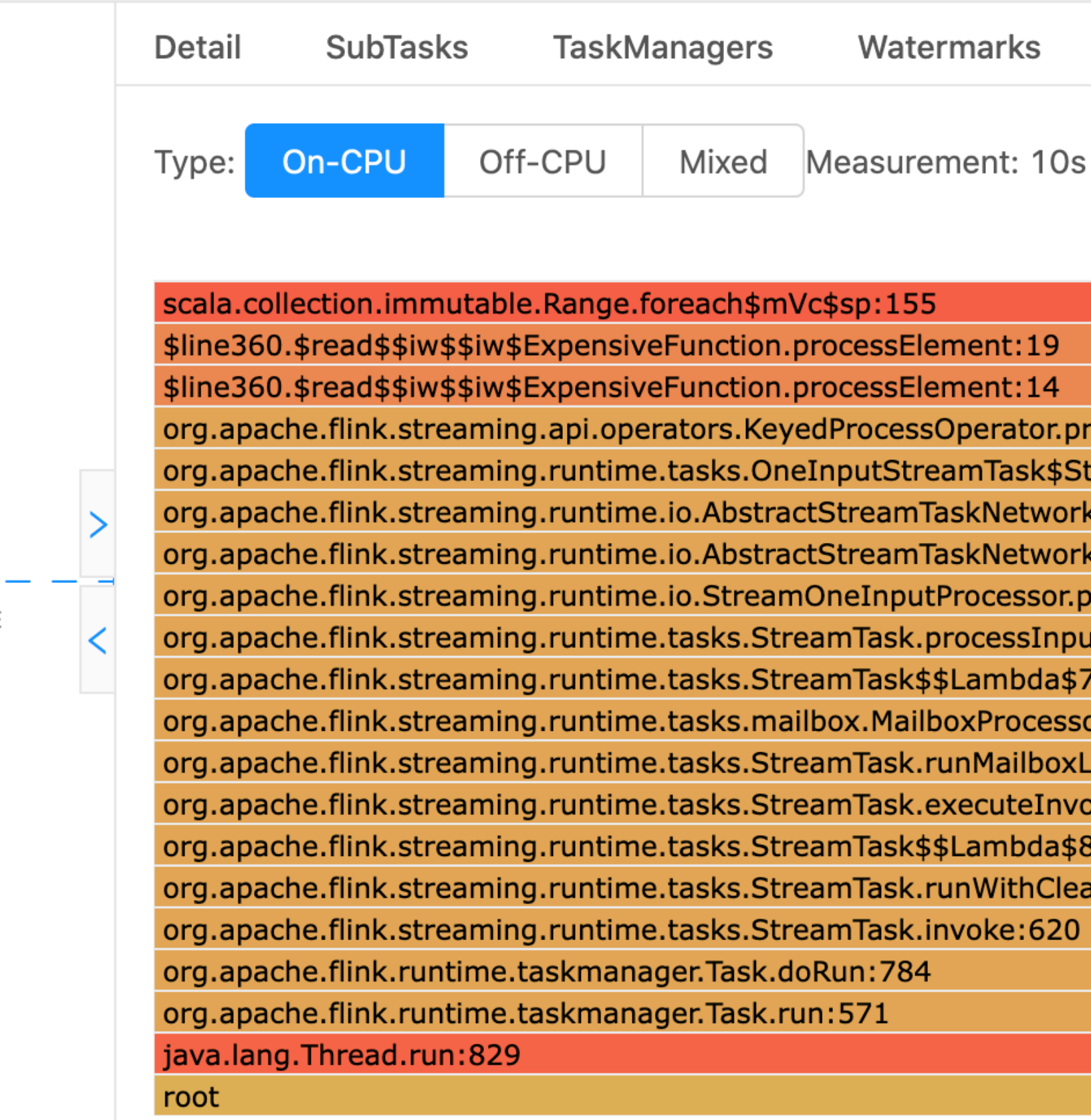
如果有多个同名的运算符，这可能会变得令人困惑，但我们可以命名运算符来解决这个问题。例如：

```
....
.process(new ExpensiveFunction).name("Expensive function")
```

火焰图

Flame graph 是一种有用的调试工具，它可以可视化目标代码的堆栈轨迹，从而可以识别最常见的代码路径。它们是通过对堆栈跟踪进行多次采样而创建的。火焰图的 x 轴显示不同的堆栈配置文件，而 y 轴显示堆栈深度和堆栈跟踪中的调用。火焰图中的单个矩形代表堆栈框架，帧的宽度显示了它在堆栈中出现的频率。有关火焰图及其使用方法的更多详细信息，请参阅火焰图表。

在 Flink 中，可以通过 Web UI 访问操作员的火焰图，方法是选择一个运算符，然后选择FlameGraph选项卡。采集到足够的样本后，将显示火焰图。以下是花 FlameGraph ProcessFunction 了很多时间才能检查的内容。


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，显示所有的 CPU 时间都花在了 ExpensiveFunction 操作员的 foreach 视图中。processElement您还会获得行号，以帮助确定代码在何处执行。

检查点已超时

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。本节介绍这种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，则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）。检查您的应用程序日志和指标是否存在以下症状：

- 您的代码中有错误。
- 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您的应用程序数据，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 Apache Flink 文档中的自定义序列化器。
- 内存不足错误。
- 以下指标出现峰值或稳定增长：
  - heapMemoryUtilization
  - oldGenerationGCTime
  - oldGenerationGCCount
  - lastCheckpointSize
  - lastCheckpointDuration

有关监控检查点的更多信息，请参阅 Apache Flink 文档中的监控检查点。

原因和解决方法

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. 33)。
- 您的应用程序状态大小太大。您可以使用该lastCheckpointSize指标监控应用程序的状态大小。
- 您的应用程序的状态数据在密钥之间分布不均匀。如果您的应用程序使用KeyBy运算符，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，则会造成瓶颈，从而导致故障。
- 您的应用程序遇到内存或垃圾收集背压。监控应用程序的heapMemoryUtilizationoldGenerationGCTime、和是否出现峰oldGenerationGCCount值或稳步增加的值。

Apache Beam 应用程序的检查点失败

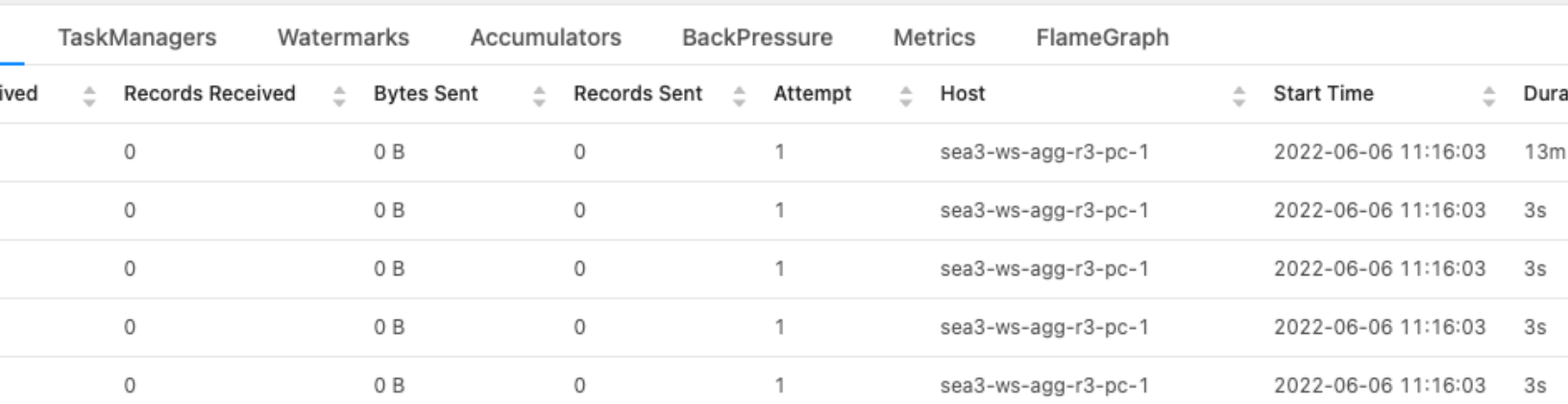
如果您的 Beam 应用程序配置为shutdownSourcesAfterIdleMs设置为 0ms，则检查点可能无法触发，因为任务处于“完成”状态。本节介绍这种情况的症状和解决方法。

症状

转到您的 Kinesis Data Analytics 应用程序 CloudWatch 日志，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，检查点未能触发。

```
               {
               "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 “完成” 状态，不再可能进行检查点了。



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

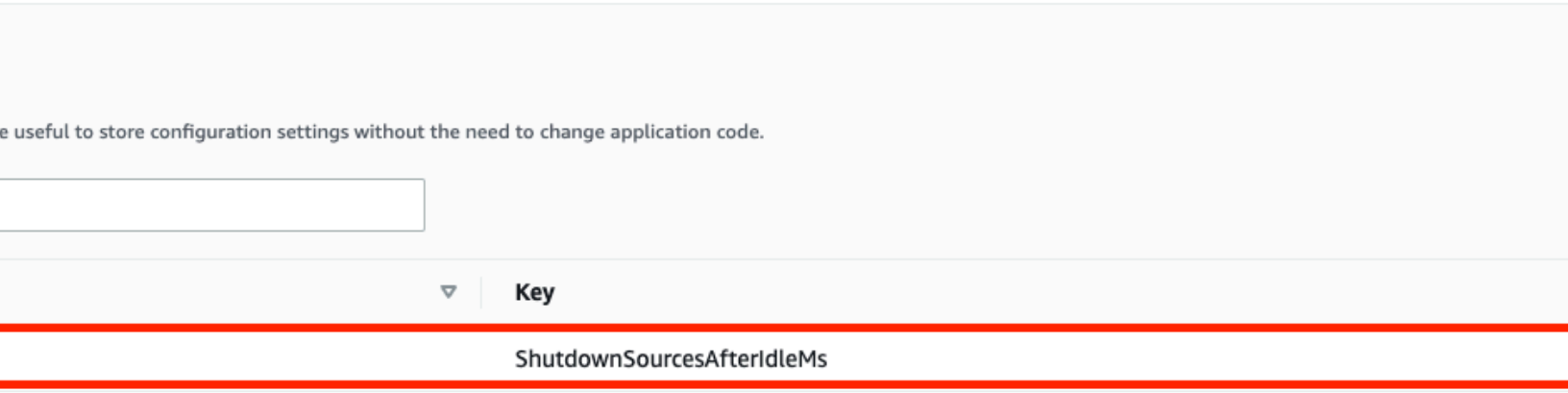
shutdownSourcesAfterIdleMs 是一个 Beam 配置变量，用于关闭在配置的毫秒时间内处于空闲状态的信号源。一旦信号源被关闭，就无法再进行检查点了。这可能导致检查点失败。

任务进入 “完成” 状态的原因之一 shutdownSourcesAfterIdleMs 是设置为 0 毫秒时，这意味着空闲的任务将立即关闭。

解决方案

要防止任务立即进入 “完成” 状态，请设置为 shutdownSourcesAfterIdleMs long.max_Value。可以通过以下两种方式执行此操作：

- 选项 1：如果您的光束配置是在 Kinesis Data Analytics 应用程序配置页面中设置的，则可以添加新的键值对来设置 shutdpwnSourcesAfteridle Ms，如下所示：



- 选项 2：如果在 JAR 文件中设置了光束配置，则可以按 shutdownSourcesAfterIdleMs 如下方式进行设置：

```
                        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。

操作员可能难以继续处理收到的消息量，原因有很多。该操作可能需要的 CPU 资源可能超过操作员可用的 CPU 资源，操作员可能会等待 I/O 操作完成。如果操作员处理事件的速度不够快，就会在上游操作员中产生背压，从而向慢速操作员提供反压。这会导致上游操作员减速，这会进一步将背压传播到源头，并通过减速来使源代码适应应用程序的整体吞吐量。你可以在 Apache Flink™ 如何处理背压中找到对背压及其工作原理的更深入的描述。

了解应用程序中哪些运算符运行缓慢，可以为您提供了解应用程序性能问题的根本原因的关键信息。背压信息通过 Flink 控制面板公开。要识别慢速运算符，请寻找背压值较高且最接近汇点的运算符（以下示例中的运算符 B）。因此，导致慢速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，但会受到反压，因为它无法将输出转发给实际的慢速操作员 C。

```
A (backpressured 93%) -> B (backpressured 85%) -> C (backpressured 11%) -> D (backpressured
 0%)
```

一旦你确定了慢速运算符，试着理解为什么它很慢。原因可能有很多，有时问题并不明显，可能需要数天的调试和分析才能解决。以下是一些明显且更常见的原因，其中一些将在下面进一步解释：

- 操作员正在进行缓慢的 I/O，例如网络调用（考虑改用 AsyncIO）。
- 数据存在偏差，一个操作员接收的事件比其他操作员多（通过查看 Flink 仪表板中各个子任务（即同一个操作员的实例）的传入/传出消息数量进行验证。

- 这是一项资源密集型操作（如果不存在数据偏差，可以考虑向外扩展到 CPU/内存密集型工作，或者增加对 I/O 限制的工作`ParallelismPerKPU`进行扩展）
- 操作员大量登录（将生产应用程序的日志记录降至最低，或者考虑改为将调试输出发送到数据流）。

使用丢弃接收器测试吞吐量

丢弃 Sink 在执行应用程序时只会忽略接收到的所有事件（没有任何 sink 的应用程序无法执行）。这对于吞吐量测试、概要分析以及验证应用程序是否正常扩展非常有用。这也是一项非常实用的理智检查，用于验证水槽是否造成了背压或应用（但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，您还可以提高并行度，以验证应用程序是否可以正常扩展，并且不会出现只有在更高的吞吐量时才会出现的瓶颈（例如，由于数据偏差）。

Data 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 使用了键控流的概念，这本质上意味着数据流的事件根据特定的密钥（例如客户 ID）进行分区，然后 Flink 可以处理不同节点上的不同分区。然后根据这些分区对许多 Flink 运算符进行评估，例如，键控 Windows、Proc es s Functions 和 Async I/O。

选择分区键通常取决于业务逻辑。同时，许多最佳实践，例如 DynamoDB 和 Spark，同样适用于 Flink，包括：

- 确保分区键的高基数
- 避免分区之间事件音量出现偏差

您可以通过比较 Flink 控制面板中接收/发送的子任务（即同一操作员的实例）的记录来识别分区中的偏差。此外，还可以将 Kinesis Data Analytics 监控配置为显示子任务级别`numRecordsIn/Out`和`numRecordsInPerSecond/OutPerSecond`子任务级别的指标。

状态偏斜

对于有状态的运算符，即为其业务逻辑（例如窗口）维护状态的运算符，数据偏差总是会导致状态偏差。由于数据偏差，一些子任务比其他子任务接收更多的事件，因此还会将更多的数据保留在状态中。但是，即使对于分区均衡的应用程序，状态中保留的数据量也可能存在偏差。例如，对于会话窗口，某些用户和会话分别可能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，则可能导致同一操作员的不同子任务所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，操作员状态将持久保存到 Amazon S3，以保护状态免受节点或集群故障的影响。在此过程中（尤其是在默认情况下在 Kinesis Data Analytics 上仅启用一次语义的情况下），从外部角度来看，处理会停止，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会受到单个子任务的限制，该子任务积累了特别多的状态。在极端情况下，由于单个子任务无法保持状态，获取检查点/保存点可能会失败。

因此，与数据偏差类似，状态偏差会大大减慢应用程序的速度。

要识别状态偏差，可以利用 Flink 仪表板。查找最近的检查点或保存点，并在详细信息中比较为单个子任务存储的数据量。

整合不同地区的资源

您可以通过 Flink 配置中的跨区域复制所需的设置，启用写入与 Kinesis Data Analytics 应用程序不同区域的 Amazon S3 存储桶。`StreamingFileSink`为此，请在Amazon Web Services Support中心提交支持请求。

Apache Amazon Kinesis Data Analytics for Apache

下表描述了自 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
- 文档最近更新时间：2022 年 11 月 22 日

更改	说明	日期
Support Apache Flink 1.15.2	Kinesis Data Analytics 现在支持使用 Apache Flink 版本 1.15.2 的应用程序。使用 Apache Flink Table 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)：	2022 年 11 月 22 日
Support Apache Flink 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 Table API 创建 Kinesis Data Analytics 应用程序。有关更多信息，请参阅入门：Flink 1.13.2 (p. 326)：	2021 年 10 月 13 日
Support Python	Kinesis Data Analytics 现在支持使用 Python 和 Apache Flink Table API 和 SQL 的应用程序。有关更多信息，请参阅使用 Python (p. 21)：	2021 年 3 月 25 日
Support Apache Flink 1.11.1	Kinesis Data Analytics 现在支持使用 Apache Flink 1.11.1 的应用程序。使用 Apache Flink Table 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 控制板	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Amaze Flink 控制面板 (p. 44)：	2020 年 11 月 19 日
EFO 消费者	创建使用增强型扇出 (EFO) 消费者从 Kinesis 数据流中读取数据的应用程序。有关更多信息，请参阅EFO 消费者 (p. 171)：	2020 年 10 月 6 日
Apache Be	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 CloudFormation 与 Kinesis Data Analytics (p. 136)：	2020 年 9 月 15 日

更改	说明	日期
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. 301)：	2020 年 7 月 21 日
自定义密钥库	如何访问使用自定义密钥库进行传输中加密的 Amazon MSK 集群。有关更多信息，请参阅自定义信任库 (p. 194)：	2020 年 6 月 10 日
CloudWatch 警报	使用 Kinesis Data Analytics for Apache Flink 创建 CloudWatch 警报的建议 有关更多信息，请参阅告警 (p. 290)：	2020 年 6 月 5 日
新 CloudWatch 指标	Kinesis Data Analytics 现在向亚马逊指标发送 22 个 CloudWatch 指标。有关更多信息，请参阅指标与维度 (p. 278)：	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到亚马逊 CloudWatch 指标。有关更多信息，请参阅自定义 指标 (p. 287)：	2020 年 5 月 12 日
示例：从不同账户的 Kinesis 流中读取	了解如何在 Kinesis 数据分析应用程序中的其他Amazon账户中访问 Kinesis 直播。有关更多信息，请参阅跨账户 (p. 188)：	2020 年 3 月 30 日
支持 Apache Flink 1.8.2	Kinesis Data Analytics 现在支持使用 Apache Flink 1.8.2 的应用程序。使用 Flink StreamingFileSink 连接器将输出直接写入 S3。有关更多信息，请参阅创建应用程序 (p. 3)：	2019 年 12 月 17 日
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，请参阅使用亚马逊 VPC (p. 392)：	2019 年 11 月 25 日
Kinesis Data Analytics s	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. 310)：	2019 年 10 月 14 日
Kinesis Data Analytics 应用程序日志	使用 CloudWatch Logs InsiKinesis Data Analytics 应用程序。有关更多信息，请参阅分析日志 (p. 275)：	2019 年 6 月 26 日
Kinesis Data Analytics 应用程序	在 Data Analytics for Apache Flink 中使用运行时属性。有关更多信息，请参阅运行时属性 (p. 25)：	2019 年 6 月 24 日

更改	说明	日期
为 Kinesis Data Analytics	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. 36)：	2019 年 5 月 8 日
Kinesis Data Analytics s	Amazon Kinesis Data Analytics 的示例应用程序演示了窗口操作员和将输出写入 CloudWatch 日志。有关更多信息，请参阅示例 (p. 146)：	2019 年 5 月 1 日
使用 Kinesis Data Analytics API 调用Amazon CloudTrail	Azon Kinesis Data Analytics 与 Amazon CloudTrail集成，后者是在 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。有关更多信息，请参阅使用 Amazon CloudTrail (p. 298)：	2019 年 3 月 22 日
创建应用程序（Kinesis Data Firehose Sink）	练习为 Apache Flink 创建 Amazon Kinesis Data Analytics，将 Amazon Kinesis 数据流作为来源，亚马逊 Kinesis Data Firehose 交付流作为汇点。有关更多信息，请参阅Kinesis Data Firehose (p. 178)：	2018 年 12 月 13 日
公开发行版	这是 Kinesis Data Analytics for Java Analytics 开发指南的初始版本。	2018 年 11 月 27 日

Kinesis Data Analytics

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
```

有关将 JSON 与 JSON 一起使用的更多信息Amazon CLI，请参阅Amazon Command Line Interface用户指南中的生成 CLI 框架和 CLI 输入 JSON 参数。

主题

AddApplicationCloudWatchLoggingOption

以下AddApplicationCloudWatchLoggingOption操作请求代码示例向 Kinesis Data Analytics 应用程序添加了 Amazon CloudWatch 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
}
```

AddApplicationInput

以下AddApplicationInput操作的示例请求代码将应用程序输入添加到 Kinesis Data Analytics 应用程序：

```
{
   "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"
      }
   }
}
```

AddApplicationInputProcessingConfiguration

以下AddApplicationInputProcessingConfiguration操作的示例请求代码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"
      }
   }
}
```

AddApplicationOutput

以下AddApplicationOutput操作的示例请求代码将 Kinesis 数据流作为应用程序输出添加到 Kinesis Data Analytics 应用程序：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "Output": {
      "DestinationSchema": {
         "RecordFormatType": "JSON"
      },
      "KinesisStreamsOutput": {
         "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/ExampleOutputStream"
      },
      "Name": "DESTINATION_SQL_STREAM"
   }
}
```

AddApplicationReferenceDataSource

以下AddApplicationReferenceDataSource操作的示例请求代码将 CSV 应用程序参考数据源添加到 Kinesis Data Analytics 应用程序：

```
{
   "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

以下CreateApplication操作的示例请求代码创建 Kinesis Data Analytics 应用程序：

```
{
  "ApplicationName":"MyApplication",
  "ApplicationDescription":"My-Application-Description",
  "RuntimeEnvironment":"FLINK-1_15",
  "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
      }
    }
  }
}
```

CreateApplicationSnapshot

以下CreateApplicationSnapshot操作的示例请求代码创建了应用程序状态的快照：

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

DeleteApplication

以下DeleteApplication操作的示例请求代码删除 Kinesis Data Analytics 应用程序：

```
{"ApplicationName": "MyApplication",
"CreateTimestamp": 12345678912}
```

DeleteApplicationCloudWatchLoggingOption

以下DeleteApplicationCloudWatchLoggingOption操作的示例请求代码从 Kinesis Data Analytics 应用程序中删除了 Amazon CloudWatch 日志选项：

```
{
    "ApplicationName": "MyApplication",
    "CloudWatchLoggingOptionId": "3.1"
    "CurrentApplicationVersionId": 3
}
```

DeleteApplicationInputProcessingConfiguration

以下DeleteApplicationInputProcessingConfiguration操作的示例请求代码从 Kinesis Data Analytics 应用程序中移除了输入处理配置：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "InputId": "2.1"
}
```

DeleteApplicationOutput

以下DeleteApplicationOutput操作的示例请求代码从 Kinesis Data Analytics 应用程序中移除了应用程序输出：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "OutputId": "4.1"
}
```

DeleteApplicationReferenceDataSource

以下DeleteApplicationReferenceDataSource操作的示例请求代码从 Kinesis Data Analytics 应用程序中移除应用程序参考数据源：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5,
   "ReferenceId": "5.1"
}
```

DeleteApplicationSnapshot

以下DeleteApplicationSnapshot操作的示例请求代码删除了应用程序状态的快照：

```
{
   "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

以下DescribeApplication操作的示例请求代码返回有关 Kinesis Data Analytics 应用程序的详细信息：

```
{"ApplicationName": "MyApplication"}
```

DescribeApplicationSnapshot

以下DescribeApplicationSnapshot操作的示例请求代码返回有关应用程序状态快照的详细信息：

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

DiscoverInputSchema

以下DiscoverInputSchema操作的示例请求代码从流式传输源生成架构：

```
{
   "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"
}
```

以下DiscoverInputSchema操作的示例请求代码从参考源生成架构：

```
{
   "S3Configuration": {
      "BucketARN": "arn:aws:s3:::mybucket",
      "FileKey": "TickerReference.csv"
   },
   "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

ListApplications

以下ListApplications操作的示例请求代码会返回您账户中的 Kinesis Data Analytics 应用程序列表：

```
{
```

```
   "ExclusiveStartApplicationName": "MyApplication",
   "Limit": 50
}
```

ListApplicationSnapshots

以下ListApplicationSnapshots操作的示例请求代码返回应用程序状态快照列表：

```
{"ApplicationName": "MyApplication",
   "Limit": 50,
   "NextToken": "aBcDeFgHiJkLmNoPqRsTuVwXyZ0123"
}
```

StartApplication

以下StartApplication操作的示例请求代码启动 Kinesis Data Analytics 应用程序，并从最新快照（如果有）加载应用程序状态：

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

StopApplication

以下 API_StopApplication 操作的示例请求代码可停止 Kinesis Data Analytics 应用程序：

```
{"ApplicationName": "MyApplication"}
```

UpdateApplication

以下UpdateApplication操作的示例请求代码更新 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 for Apache Flink

有关 Kinesis Data Analytics for Apache Flink 提供的 API 的信息，请参阅 Kinesis Data Analytics API 参考资料。

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