Managed Service for Apache Flink 開發人員指南

Managed Service for Apache Flink

Managed Service for Apache Flink: Managed Service for Apache Flink 開發人員指南

Table of Contents

Amazon Managed Service for Apache Flink 之前被稱為 Amazon Kinesis Data Analytics for Apache Flink。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。

什麼是 Amazon Managed Service for Apache Flink？

透過 Amazon 阿帕奇 Flink 受管服務，您可以使用 Java、斯卡拉、Python 或 SQL 來處理和分析串流資料。此服務可讓您針對串流來源和靜態來源撰寫和執行程式碼，以執行時間序列分析、饋送即時儀表板和指標。

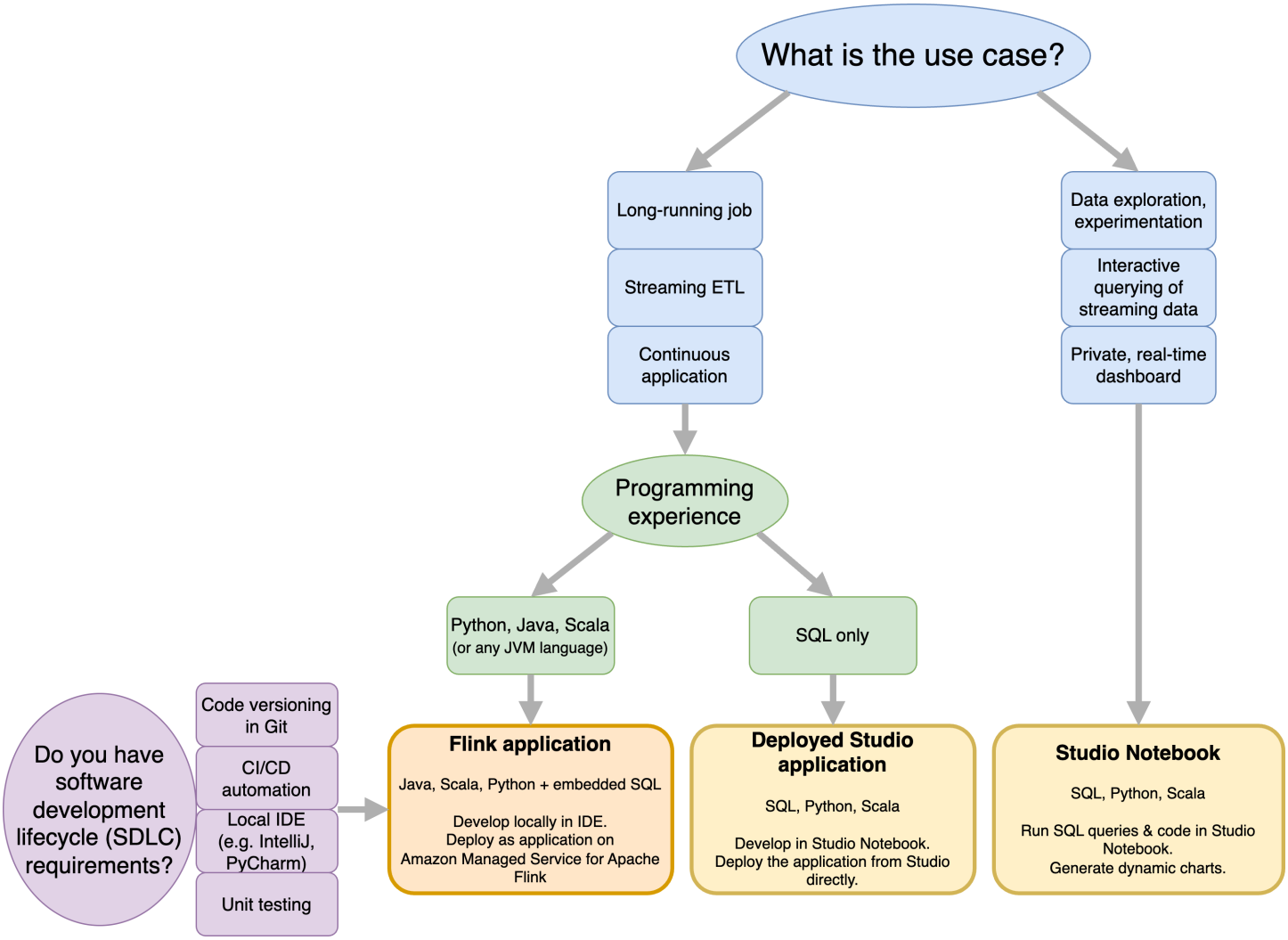
您可以使用以 Apache Flink 為基礎的開放原始碼程式庫，在 Apache Flink 的受管理服務中，使用您選擇的語言來建置應用程式。Apache Flink 是處理資料串流的熱門框架及引擎。

Managed Service for Apache Flink 可為 Apache Flink 應用程式提供基礎設施。它可處理核心功能，例如佈建運算資源、AZ 容錯移轉復原能力、parallel 運算、自動調整規模，以及應用程式備份 (以檢查點和快照實作)。您可以使用高階 Flink 程式設計功能 (例如運算子、函數、來源和接收器)，使用方式與您自行託管 Flink 基礎架構時相同。

選擇阿帕奇 Flink 管理服務或管理服務的阿帕奇 Flink 工作室

您有兩個選項可以使用適用於 Apache Flink 的 Amazon 受管服務來執行 Flink 任務。透過適用於 Apache Flink 的受管理服務，您可以使用您選擇的 IDE 和 Apache Flink 資料流或資料表 API，以 Java、斯卡拉或 Python (以及嵌入式 SQL) 建置 Flink 應用程式。透過適用於 Apache Flink Studio 的受管理服務，您可以即時以互動方式查詢資料串流，並使用標準 SQL、Python 和 Scala 輕鬆建置和執行串流處理應用程式。

您可以選取最適合您使用案例的方法。如果您不確定，本節將提供高級指導來幫助您。



在決定是否為 Apache Flink 使用 Amazon 託管服務還是阿帕奇 Flink 工作室的 Amazon 託管服務之前，您應該考慮使用案例。

如果您打算操作一個長時間運行的應用程序，該應用程序將承擔諸如流 ETL 或連續應用程序之類的工作負載，則應考慮使用 Apache Flink 的託管服務。這是因為您可以直接在您選擇的 IDE 中使用 Flink API 建立您的 Flink 應用程式。使用 IDE 在本機開發也可確保您可以利用軟體開發生命週期 (SDLC) 常見程序和工具，例如 Git、CI/CD 自動化或單元測試中的程式碼版本控制。

如果您對臨時資料探索感興趣、想要以互動方式查詢串流資料或建立私人即時儀表板，Apache Flink Studio 的受管服務將協助您只要按幾下滑鼠即可達成這些目標。熟悉 SQL 的使用者可以考慮直接從 Studio 部署長時間執行的應用程式。

> **Note**
>
> 您可以將您的 Studio 筆記本升級為長時間運行的應用程序。不過，如果您想要與 SDLC 工具整合，例如 Git 和 CI/CD 自動化上的程式碼版本控制，或單元測試等技術，我們建議您使用您選擇的 IDE 為 Apache Flink 管理服務。

選擇要在 Apache Flink 管理服務中使用的 Apache Flink 應用程式介面

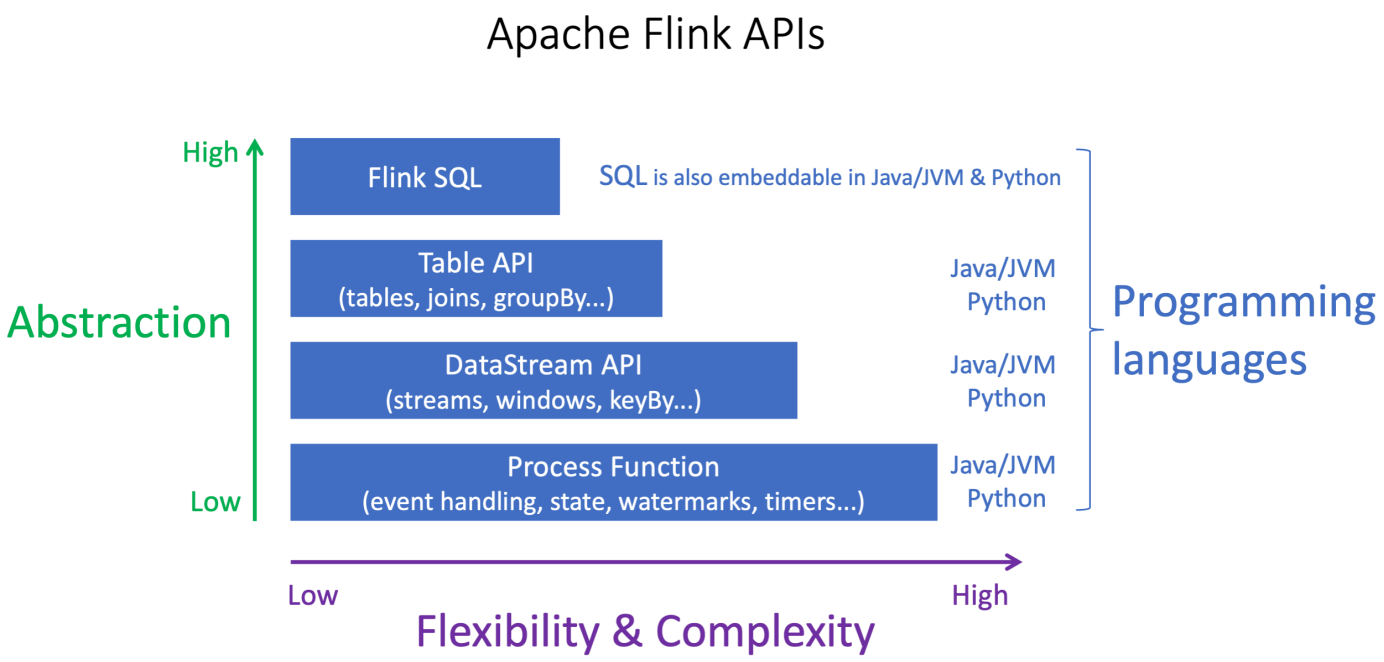
您可以在您選擇的 IDE 中使用 Apache Flink 的 API 在管理服務中使用 Java，Python 和斯卡拉構建應用程序。您可以在文件中找到有關如何使用 Flink 資料流和表格 API 建置應用程式的指引。您可以選擇建立 Flink 應用程式所使用的語言，以及最符合應用程式和作業需求的 API。如果您不確定，本節將提供高階指引以協助您。

選擇一個快速連結 API

Apache Flink API 具有不同層級的抽象概念，可能會影響您決定建置應用程式的方式。它們具有表現力和靈活性，可以一起使用來構建您的應用程序。您不需要只使用一個 Flink API。您可以在 Apache Flink 文件中進一步了解有關 Flink API 的資訊。

Flink 提供了四個級別的 API 抽象：Flink SQL，表 API， DataStream API 和過程函數，這是與 API 一起使用。 DataStream 這些都在阿帕奇 Flink 的 Amazon 託管服務中受到支持。建議盡可能從更高級別的抽象開始，但是某些 Flink 功能僅適用於數據流 API，您可以在其中使用 Java，Python 或 Scala 創建應用程序。在下列情況下，您應該考慮使用資料流 API：

- 您需要對狀態進行細粒度控制
- 您希望利用異步調用外部數據庫或端點的能力（例如推斷）
- 您想要使用自訂計時器



> **Note**
>
> 使用資料流 API 選擇語言：
>
> - SQL 可以嵌入任何 Flink 應用程式中，無論選擇的程式設計語言為何。
> - 如果您打算使用 DataStream API，則 Python 中並非所有連接器都受到支援。
> - 如果您需要低延遲/高吞吐量，則無論 API 如何，都應該考慮使用 Java/Scala。
> - 如果您打算在流程函數 API 中使用異步 IO，則需要使用 Java。

開始

您可以先建立可持續讀取和處理串流資料的 Managed Service for Apache Flink 應用程式。然後，使用您選擇的 IDE 編寫程式碼，並使用即時串流資料對其進行測試。您也可以設定希望 Managed Service for Apache Flink 傳送結果的目的地。

建議您閱讀下列章節入門：

- Managed Service for Apache Flink：如何運作
- 開始使用適用於阿帕奇 Flink 的 Amazon 託管服務 (DataStream API)

另一方面，您可以先建立 Apache Flink Studio 筆記本的受管理服務，讓您即時以互動方式查詢資料串流，並使用標準 SQL、Python 和 Scala 輕鬆建置和執行串流處理應用程式。只要在 AWS Management Console 按幾下滑鼠，即可啟動無伺服器筆記本，以查詢資料串流並在幾秒鐘內取得結果。建議您閱讀下列章節入門：

- 將 Studio 筆記本用於 Managed Service for Apache Flink
- 建立 Studio 筆記本

Managed Service for Apache Flink：如何運作

Amazon Managed Service for Apache Flink 是全受管 Amazon 服務，可讓您建立和管理 Apache Flink 應用程式以處理串流資料。

Apache Flink 應用程式的程式設計

Apache Flink 應用程式是使用 Apache Flink 框架建立的 Java 或 Scala 應用程式。您可以在本機編寫並建置 Apache Flink 應用程式。

應用程式主要使用 DataStream API 或資料表 API。您也可以使用其他 Apache Flink API，但是這些 API 在建置串流應用程式時較不常用。

兩種 API 的功能如下：

DataStream API

Apache Flink DataStream API 程式設計模型以兩個元件為基礎：

- 資料串流：資料記錄之連續資料流的結構化表示。
- 轉換運算子：接受一或多個資料串流作為輸入，並產生一或多個資料串流作為輸出。

使用 DataStream API 建立的應用程式會執行下列動作：

- 從資料來源 (例如 Kinesis 串流或 Amazon MSK 主題) 讀取資料。
- 將轉換套用至資料，例如篩選、彙總或富集。
- 將轉換後的資料寫入資料接收器。

使用 DataStream API 的應用程式可以使用 Java 或 Scala 編寫，而且可以從 Kinesis 資料串流、Amazon MSK 主題或自訂來源讀取。

應用程式使用連接器來處理資料。Apache Flink 使用以下類型的連接器：

- 來源：用於讀取外部資料的連接器。
- 接收器：用於寫入外部位置的連接器。
- 運算子：用於處理應用程式內資料的連接器。

典型的應用程式包含至少一個具有來源的資料串流、具有一或多個運算子的資料串流，以及至少一個資料接收器。

如需如何使用 DataStream API 的詳細資訊，請參閱 DataStream API。

資料表 API

Apache Flink 資料表 API 程式設計模型以以下元件為基礎：

- 資料表環境：用於建立和託管一個或多個資料表的基礎資料的介面。
- 資料表：提供 SQL 資料表或檢視存取權的物件。
- 資料表來源：用於從外部來源 (例如 Amazon MSK 主題) 讀取資料。
- 資料表函數：用於轉換資料的 SQL 查詢或 API 呼叫。
- 資料表接收器：用於將資料寫入外部位置，例如 Amazon S3 儲存貯體。

使用資料表 API 建立的應用程式會執行下列動作：

- 透過連線至 `Table Source` 建立 `TableEnvironment`。
- 使用 SQL 查詢或資料表 API 函數在 `TableEnvironment` 中建立資料表。
- 使用資料表 API 或 SQL 在資料表上執行查詢。
- 使用資料表函數或 SQL 查詢對查詢結果套用轉換。
- 將查詢或函數結果寫入 `Table Sink`。

使用資料表 API 的應用程式可以用 Java 或 Scala 編寫，並且可以使用 API 調用或 SQL 查詢查詢資料。

如需如何使用資料表 API 的詳細資訊，請參閱 資料表 API。

建立 Managed Service for Apache Flink 應用程式

Managed Service for Apache Flink 是一項 AWS 服務，可建立託管 Apache Flink 應用程式的環境，並為其提供下列設定：

- 執行時間屬性：可以提供給應用程式的參數。您可以變更這些參數，無需重新編譯應用程式的程式碼。

- 容錯能力：應用程式如何從中斷和重新啟動中復原。
- 記錄和監控：應用程式如何將事件記錄到 CloudWatch Logs。
- 擴展：應用程式如何佈建運算資源。

您可以使用主控台或 AWS CLI 建立 Managed Service for Apache Flink 應用程式。若要開始建立 Managed Service for Apache Flink 應用程式，請參閱開始使DataStream 用 (API)。

建立 Managed Service for Apache Flink 應用程式

本主題包含如何建立 Managed Service in Apache Flink 的相關資訊。

本主題包含下列章節：

- 建置 Managed Service in Apache Flink 應用程式的程式碼
- 建立 Managed Service for Apache Flink 應用程式
- 啟動 Managed Service for Apache Flink 應用程式
- 驗證 Managed Service for Apache Flink 應用程式

建置 Managed Service in Apache Flink 應用程式的程式碼

本節說明您用來建置 Managed Service for Apache Flink 應用程式程式碼的元件。

建議將 Apache Flink 應用程式的最新支援版本用於您的應用程式程式碼。Managed Service for Apache Flink 支援的最新 Apache Flink 版本是 1.15.2。如需升級 Managed Service for Apache Flink 應用程式的相關資訊，請參閱升級應用程式。

您可以使用 Apache Maven 建置應用程式的程式碼。Apache Maven 專案使用 `pom.xml` 檔案來指定它使用的元件的版本。

> **Note**
>
> Managed Service for Apache Flink 支援最大 512 MB 的 JAR 檔案。如果使用的 JAR 檔案大於此大小，應用程式將無法啟動。

將下列元件版本用於 Managed Service for Apache Flink 應用程式：

元件	版本
Java	11 (建議使用)
Scala	請參閱下面的 Scala 解耦註釋
阿帕奇 Flink 運行時的託管服務（ ）aws-kinesisanalytics-runtime	1.2.0
AWSKinesis 連接器 () flink-connector-kinesis	1.15.2
Apache Beam (僅限於 Beam 應用程式)	2.33.0，帶有 Jackson 2.12.2 版

從版本 1.15 開始，Flink 中移除了 Scala 相依性。應用程式現在可以使用任何 Scala 版本的 Java API。您需要將您選擇的 Scala 標準程式庫綁定到 Scala 應用程式中。

如需使用 Apache Flink 1.15.2 版的 Managed Service for Apache Flink 的 `pom.xml` 檔案範例，請參閱 Managed Service for Apache Flink 入門。

如需建立使用 Apache Beam 之 Managed Service for Apache Flink 應用程式的相關資訊，請參閱使用 Apache Beam。

指定應用程式的 Apache Flink 版本

使用 Managed Service for Apache Flink 執行時間 1.1.0 版及更新版本時，您可以指定您編譯應用程式時應用程式使用的 Apache Flink 版本。您可以使用 `-Dflink.version` 參數提供 Apache Flink 的版本，如下所示：

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

若要使用較舊版本的 Apache Flink 建置應用程式，請參閱舊版本。

建立 Managed Service for Apache Flink 應用程式

建置應用程式的程式碼之後，請執行下列動作來建立 Managed Service for Apache Flink 應用程式：

- 上傳應用程式程式碼：將應用程式的程式碼上傳至 Amazon S3 儲存貯體。建立應用程式時，請指定應用程式程式碼的 S3 儲存貯體名稱和物件名稱。如需說明如何上傳應用程式程式碼的教學課程，請

參閱 開始使DataStream 用 (API) 教學課程中的the section called “上傳 Apache Flink 串流 Java 程式碼”。

- 建立 Managed Service for Apache Flink：使用下列其中一種方法建立 Managed Service for Apache Flink 應用程式：
  - 使用 AWS 主控台建立 Managed Service for Apache Flink 應用程式：您可以使用 AWS 主控台建立和設定應用程式。

    使用主控台建立應用程式時，系統會為您建立應用程式的相依資源 (例如 CloudWatch 日誌串流、IAM 角色和 IAM 政策)。

    使用主控台建立應用程式時，您可以從 Managed Service for Apache Flink -建立應用程式頁面的下拉式清單中選取版本，來指定應用程式使用的 Apache Flink 版本。

    如需如何使用主控台建立應用程式的教學課程，請參閱 開始使DataStream 用 (API) 教學課程the section called “建立和執行應用程式 (主控台)”中的。
  - 使用 AWS CLI 建立 Managed Service for Apache Flink 應用程式：您可以使用 AWS CLI 建立和設定應用程式。

    使用 CLI 建立應用程式時，還必須手動建立應用程式的相依資源 (例如 CloudWatch 日誌串流、IAM 角色和 IAM 政策)。

    使用 CLI 建立應用程式時，您可以使用 `CreateApplication` 動作的 `RuntimeEnvironment` 參數指定應用程式使用的 Apache Flink 版本。

    如需如何使用 CLI 建立應用程式的教學課程，請參閱 開始使DataStream 用 (API) 教學課程中的the section called “使用 CLI 建立和執行應用程式”。

> **Note**
>
> 您無法變更現有應用程式的`RuntimeEnvironment`。如果您需要變更現有應用程式的`RuntimeEnvironment`，則必須刪除該應用程式並重新建立。

啟動 Managed Service for Apache Flink 應用程式

建置應用程式程式碼、將程式碼上傳至 S3，並建立 Managed Service for Apache Flink 應用程式之後，即可啟動應用程式。啟動 Managed Service Apache Flink 應用程式通常需要幾分鐘時間。

使用下列其中一種方法來啟動應用程式：

- 使用 AWS 主控台啟動 Managed Service for Apache Flink：您可以在 AWS 主控台的應用程式頁面上選擇執行來執行應用程式。
- 使用 AWS API 啟動適用於 Apache Flink 應用程式的受管理服務：您可以使用StartApplication動作執行應用程式。

驗證 Managed Service for Apache Flink 應用程式

您可以驗證應用程式是否正常運作，方式如下：

- 使用 CloudWatch 日誌：您可以使用 CloudWatch 日誌和 CloudWatch 日誌洞察來驗證您的應用程序是否正常運行。如需將 CloudWatch 記錄與 Apache Flink 應用程式的受管理服務搭配使用的相關資訊，請參閱記錄和監控。
- 使用 CloudWatch 指標：您可以使用 CloudWatch 指標來監控應用程式的活動，或應用程式用於輸入或輸出的資源中的活動 (例如 Kinesis 串流、Kinesis Data Firehose 串流或 Amazon S3 儲存貯體)。如需有關指 CloudWatch 標的詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的使用指標。
- 監控輸出位置：如果應用程式將輸出寫入某個位置 (例如 Amazon S3 儲存貯體或資料庫)，您可以為寫入的資料監控該位置。

執行 Managed Service for Apache Flink 應用程式

本主題包含執行 Managed Service for Apache Flink 的相關資訊。

執行 Managed Service for Apache Flink 應用程式時，該服務會建立 Apache Flink 作業。Apache Flink 作業是指 Managed Service in Apache Flink 應用程式的執行生命週期。作業的執行及其使用的資源由作業管理員管理。作業管理員會將應用程式的執行分隔為任務。每個任務由任務管理員管理。監視應用程式的效能時，您可以檢查每個任務管理員的效能或作業管理員的整體效能。

如需 Apache Flink 作業的相關資訊，請參閱《Apache Flink 文件》中的作業和排程。

應用程式與作業狀態

應用程式及其作業都具有目前的執行狀態：

- 應用程式狀態：您的應用程式目前的狀態，描述其執行階段。應用程式狀態包括下列幾種：
  - 穩定的應用程式狀態：您的應用程式通常會保持下列狀態，直到您變更狀態為止：
    - READY：新的或已停止的應用程式處於 READY 狀態，直到您執行它為止。
    - RUNNING：已成功啟動的應用程式處於 RUNNING 狀態。

- 暫時性應用程式狀態：處於這些狀態的應用程式通常處於轉換至其他狀態的過程中。如果應用程式已保持暫時狀態一段時間，您可以使用 StopApplication 動作停止應用程式，並將 Force 參數設定為 true。這些狀態包括下列項目：
  - STARTING：在 StartApplication 操作之後發生。應用程式正在從狀態 READY 轉換為 RUNNING 狀態。
  - STOPPING：在 StopApplication 動作之後發生。應用程式正在從狀態 RUNNING 轉換為 READY 狀態。
  - DELETING：在 Delete Application 動作之後發生。正在刪除應用程式。
  - UPDATING：在 UpdateApplication 動作之後發生。應用程式正在更新，並會轉換回 RUNNING 或 READY 狀態。
  - AUTOSCALING：應用程式的 ParallelismConfiguration AutoScalingEnabled 屬性設定為 true，而且服務正在增加應用程式的平行處理層級。當應用程式處於此狀態時，您唯一可以使用的有效 API 動作是 StopApplication 動作，且 Force 參數設定為 true。如需自動擴展的相關資訊，請參閱自動擴展。
  - FORCE_STOPPING：在呼叫 StopApplication 動作且 Force 參數設定為 true 之後發生。正在停止應用程式。應用程式正在從 STARTING、UPDATING、STOPPING 或 AUTOSCALING 狀態轉換為 READY 狀態。
  - ROLLING_BACK：在 RollbackApplication 動作之後發生。應用程式正在復原至先前的版本。應用程式正在從 UPDATING 或 AUTOSCALING 狀態轉換為 RUNNING 狀態。
  - ROLLED_BACK：成功復原應用程式後，這會變成您從中復原的版本的狀態。如需復原應用程式的相關資訊，請參閱 RollbackApplication。
  - MAINTENANCE：在 Managed Service for Apache Flink 將修補程式套用至您的應用程式時發生。如需詳細資訊，請參閱維護。

  您可以使用主控台或 DescribeApplication 動作來檢查應用程式的狀態。
- 作業狀態：當應用程式處於 RUNNING 狀態時，作業的狀態會描述其目前的執行階段。作業會以 CREATED 狀態開始，然後在啟動時繼續進行到 RUNNING 狀態。如果發生錯誤情況，應用程式會進入下列狀態：
  - 對於使用 Apache Flink 1.11 及更新版本的應用程式，會進入 RESTARTING 狀態。
  - 對於使用 Apache Flink 1.8 及更新版本的應用程式，會進入 FAILING 狀態。

  然後，應用程式會繼續進入 RESTARTING 或 FAILED 狀態，取決於作業是否可以重新啟動。

  您可以檢查應用程式的 CloudWatch 日誌看是否有狀態變更，以檢查作業的狀態。

批次工作負載

Managed Service in Apache Flink 支援執行 Apache Flink 批次工作負載。在批次作業中，當 Apache Flink 作業進入 FINISHED 狀態時，Managed Service for Apache Flink 應用程式的狀態會設定為 READY。如需關於 Flink 作業狀態的詳細資訊，請參閱作業與排程。

應用程式資源

本節說明應用程式使用的系統資源。了解 Managed Service for Apache Flink 如何佈建和使用資源，將協助您設計、建立和維護高效能且穩定的 Managed Service for Apache Flink 應用程式。

Managed Service for Apache Flink 應用程式資源

Managed Service for Apache Flink 是一項 AWS 服務，可建立託管 Apache Flink 應用程式的環境。Managed Service for Apache Flink 服務使用 Kinesis 處理單元 (KPU) 提供資源。

一個 KPU 代表下列系統資源：

- 一個 CPU 核心
- 4 GB 的記憶體，其中 1 GB 是本機記憶體，3 GB 是堆積記憶體
- 50 GB 的可用磁碟空間

KPU 在稱為任務和子任務的不同執行單元中執行應用程式。你可以將一個子任務視為等效於一個執行緒。

應用程式可用的 KPU 數量等於應用程式的 `Parallelism` 設定除以應用程式的 `ParallelismPerKPU` 設定。

如需關於應用程式平行處理層級的詳細資訊，請參閱 擴展。

Apache Flink 應用程式資源

Apache Flink 環境會使用稱為任務空位的單元為應用程式配置資源。Managed Service for Apache Flink 為應用程式配置資源時，會將一或多個 Apache Flink 任務空位指派給單一 KPU。指派給單一 KPU 的空位數量等於應用程式的 `ParallelismPerKPU` 設定。如需關於任務空位的詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的作業排程。

運算子平行處理層級

您可以設定運算子可以使用的最大子任務數目。此值稱為運算子平行處理層級。依預設，應用程式中每個運算子的平行處理層級與應用程式的平行處理層級相同。這表示依預設，應用程式中的每個運算子都可以視需要使用應用程式中所有可用的子任務。

您可以使用 `setParallelism` 方法設定應用程式中運算子的平行處理層級。使用此方法，您可以控制每個運算子一次可以使用的子任務數目。

如需關於運算子鏈結的詳細資訊，請參閱《Apache F link 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的任務鏈結和資源群組。

運算子鏈結

通常，每個運算子都使用單獨的子任務來執行，但是如果多個運算子始終按順序執行，則執行時間可以將它們全部分配給相同的任務。這個過程稱為運算子鏈結。

如果幾個循序運算子都對相同的資料進行操作，則可以鏈接到單個任務中。以下是實現這一點所需要的一些條件：

- 運算子執行 1 對 1 簡單轉發。
- 所有運算子具有相同的平行處理層級。

應用程式將運算子鏈結到單一子任務中時，可以節省系統資源，因為服務不需要執行網路作業，也不需要為每個運算子配置子任務。若要判斷您的應用程式是否使用運算子鏈結，請查看 Managed Service for Apache Flink 主控台中的作業圖表。應用程式中的每個頂點代表一或多個運算子。此圖表顯示已鏈結為單一頂點的運算子。

DataStream API

Apache Flink 應用程式使用 Apache Flink DataStream API 來轉換資料串流中的資料。

本節包含下列主題：

- 使用連接器在 Apache Flink 的受管理服務中使用 API 移動資料 DataStream ：這些元件可以在應用程式與外部資料來源和目的地之間移動資料。
- 在 Managed Service for Apache Flink 中使用運算子與 DataStream API 轉換資料：這些元件可以轉換或分組應用程式中的資料元素。

- 使用 DataStream API 追蹤 Managed Service in Apache Flink 中的事件：本主題說明 Managed Service for Apache Flink 如何在使用 DataStream API 時追蹤事件。

使用連接器在 Apache Flink 的受管理服務中使用 API 移動資料 DataStream

在適用於 Apache Flink DataStream API 的 Amazon 受管服務中，連接器是可將資料移入和移出 Apache Flink 應用程式的受管服務的軟體元件。連接器是靈活的整合，可讓您從檔案和目錄中讀取資料。連接器包含用於與 Amazon 服務和第三方系統互動的完整模組。

連接器包含下列類型：

- 來源：從 Kinesis 資料串流、檔案或其他資料來源向應用程式提供資料。
- 接收：將資料從應用程式傳送到 Kinesis 資料串流、Kinesis Data Firehose 串流或其他資料目的地。
- 非同步 I/O：提供對資料來源 (例如資料庫) 的非同步存取，以富集串流事件。

可用的連接器

Apache Flink 架構包含用於存取各種來源之資料的連接器。如需 Apache Flink 架構中可用連接器的相關資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的連接器。

Warning

如果應用程式在 Flink 1.6、1.8、1.11 或 1.13 上執行，且想要在中東 (阿拉伯聯合大公國)、亞太區域 (海德拉巴)、以色列 (特拉維夫)、歐洲 (蘇黎世)、亞太區域 (墨爾本 或亞太區域 (雅加達) 執行，您可能需要使用更新的連接器重建應用程式存檔或需要升級至 Flink 1.15。以下是建議的指導方針：

連接器升級

Fl 版本	使用的連接器	解析度
1. - 1.	Firehose	您的應用

Fl 版 本	使用的連接器	解析度
		程式依賴舊版 Firehose 連接器，該版本不會知道新的 AWS 區域。使用 Firehose 連接器 2.1.0 版重建應

Fl 版 本	使用的連接器	解 析 度
		用 程 式 存 檔 。 2.1.0 版

Fl 版本	使用的連接器	解析度
1.	Kinesis	您的應用程式依賴舊版 Flink Kinesis 連接器，該版本不知道新的 AWS 區域。使用 Flink Kinesis 連接

Fl 版 本	使用的連接器	解 析 度 器
		1.6.1 版 重 建 應 用 程 式 存 檔 。 https:// g ithub.com / awslabs/ amazon- ki nesis- con nector- fl ink / 樹/1.6.1

Fl 版本	使用的連接器	解析度
1.	Kinesis	您的應用程式依賴舊版 Flink Kinesis 連接器，該版本不知道新的 AWS 區域。使用 Flink Kinesis 連接

Fl 版 本	使用的連接器	解析度器
		2.4.1 版重建應用程式存檔。 https://github.com/awslabs/amazon-kinesis-connector-flink/樹/2.4.1

Fl 版 本	使用的連接器	解 析 度
1. 和 1.	Kinesis	您 的 應 用 程 式 依 賴 舊 版 Flink Kinesis 連 接 器， 該 版 本 不 知 道 新 的 AWS 區 域。 很 抱 歉，Fl ink 不 再

Fl 版 本	使用的連接器	解析度
		發行 1.6/1.13 連接器的修補程式或錯誤修正。建議使用 Flink 1.15 重建應用程式封存檔，以

Fl 版本	使用的連接器	解析度
		更新至 Flink 1.15。

將串流資料來源新增至 Managed Service for Apache Flink

Apache Flink 提供了連接器，用於從檔案、通訊端、集合和自訂來源讀取資料。在應用程式的程式碼中，您可以使用 Apache Flink 來源接收來自串流的資料。本節說明可用於 Amazon 服務的來源。

Kinesis Data Streams

此 `FlinkKinesisConsumer` 來源將來自 Amazon Kinesis 資料串流的串流資料提供給應用程式。

建立`FlinkKinesisConsumer`

以下程式碼範例示範如何建立 `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));
```

如需使用 `FlinkKinesisConsumer` 的詳細資訊，請參閱 下載並檢查 Apache Flink 串流 Java 程式碼。

建立使用增強型扇出 (EFO) 取用者的 `FlinkKinesisConsumer`

FlinkKinesisConsumer 現在支援增強型扇出 (EFO)。

如果 Kinesis 取用者使用 EFO，Kinesis Data Streams 服務會提供專屬頻寬，而不是讓取用者與其他從串流讀取的取用者共用串流的固定頻寬。

如需將 EFO 與 Kinesis 取用者搭配使用的詳細資訊，請參閱 FLIP-128：適用於 AWS Kinesis 取用者的增強型扇出。

您可以在 Kinesis 取用者上設定下列參數來啟用 EFO 取用者：

- RECORD_PUBLISHER_TYPE：將此參數設定為 EFO，以便讓應用程式使用 EFO 取用者來存取 Kinesis 資料串流資料。
- EFO_CONSUMER_NAME：將此參數設定為字串值，確保在此串流的取用者中保持唯一。在相同的 Kinesis 資料串流中重複使用取用者名稱，將導致先前使用該名稱的使用者遭到終止。

若要設定 `FlinkKinesisConsumer` 使用 EFO，請將下列參數新增至取用者：

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

如需使用 EFO 取用者之 Managed Service for Apache Flink 應用程式的範例，請參閱 增強型扇出 (EFO) 取用者。

Amazon MSK

此 `KafkaSource` 來源將來自 Amazon MSK 主題的串流資料提供給應用程式。

建立`KafkaSource`

以下程式碼範例示範如何建立 `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");
```

如需使用 `KafkaSource` 的詳細資訊，請參閱 MSK 複寫。

使用 Managed Service in Apache Flink 寫入資料

在應用程式的程式碼中，您可以使用 Apache Flink 接收器將資料從 Apache Flink 串流寫入 AWS 服務，例如 Kinesis Data Streams。

Apache Flink 為檔案、通訊端和自訂接收器提供接收器。下列是 AWS 可用的接收器：

Kinesis Data Streams

Apache Flink 在《Apache Flink 文件》中提供了 Kinesis Data Streams 連接器的相關資訊。

如需使用 Kinesis 資料串流進行輸入和輸出的應用程式範例，請參閱 開始使DataStream 用 (API)。

Amazon S3

您可以使用 Apache Flink `StreamingFileSink` 將物件寫入 Amazon S3 儲存貯體。

如需如何將物件寫入 S3 的範例，請參閱the section called “S3 接收器”。

Kinesis Data Firehose

`FlinkKinesisFirehoseProducer` 是一個可靠、可擴展的 Apache Flink 接收器，可使用 Kinesis Data Firehose 服務來儲存應用程式輸出。本節說明如何建立 Maven 專案以建立和使用 `FlinkKinesisFirehoseProducer`。

主題

- 建立FlinkKinesisFirehoseProducer
- FlinkKinesisFirehoseProducer 程式碼範例

建立`FlinkKinesisFirehoseProducer`

以下程式碼範例示範如何建立 `FlinkKinesisFirehoseProducer`：

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

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

`FlinkKinesisFirehoseProducer` 程式碼範例

下列程式碼範例示範如何建立並設定 `FlinkKinesisFirehoseProducer`，並將資料從 Apache Flink 資料串流傳送至 Kinesis Data Firehose 服務。

```
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");
 }
}
```

如需關於如何使用 Kinesis Data Firehose 接收器的完整教學課程，請參閱the section called “Kinesis Data Firehose 接收器”。

在 Managed Service to Apache Flink 中使用非同步 I/O

非同步 I/O 運算子使用外部資料來源 (例如資料庫) 來富集串流資料。Managed Service for Apache Flink 以非同步方式富集串流事件，以便批次處理請求來提高效率。

如需詳細資訊，請參閱《Apache Flink 文件》中的調整檢查點https://nightlies.apache.org/flink/flink-docs-release-1.15/。

在 Managed Service for Apache Flink 中使用運算子與 DataStream API 轉換資料

若要在 Managed Service for Apache Flink 中轉換傳入的資料，請使用 Apache Flink 運算子。Apache Flink 運算子可將一或多個資料串流轉換為新的資料串流。新的資料串流包含來自原始資料串流的修改資料。Apache Flink 提供了超過 25 個預先建置的串流處理運算子。如需詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的運算子。

本主題包含下列章節：

轉換運算子

以下是在 JSON 資料串流的其中一個欄位上進行簡單文字轉換的範例。

此程式碼會建立轉換後的資料串流。新資料串流具有與原始串流相同的資料，並在 TICKER 欄位內容後面附加 Company 字串。

```
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");
        }
    }
);
```

彙總運算子

以下是彙總運算子的範例。程式碼會建立彙總的資料串流。運算子會建立 5 秒的翻轉視窗，並傳回視窗中具有相同 TICKER 值之記錄的PRICE 值總和。

```
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;
});
```

如需使用運算子的完整程式碼範例，請參閱開始使DataStream 用 (API)。入門應用程式的來源程式碼可在 Managed Service for Apache Flink Java GitHub 儲存庫的入門中取得。

使用 DataStream API 追蹤 Managed Service in Apache Flink 中的事件

Managed Service in Apache Flink 使用下列時間戳記來追蹤事件：

- 處理時間：指正在執行相應操作的機器的系統時間。
- 事件時間：指每個個別事件在其生產設備上發生的時間。
- 擷取時間：指事件進入 Managed Service for Apache Flink 服務的時間。

您可以使用 `setStreamTimeCharacteristic` 設定串流環境使用的時間：

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

如需詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的時間戳記。

資料表 API

Apache Flink 應用程式使用 Apache Flink 資料表 API，透過關聯式模型與串流中的資料互動。您可以使用資料表 API 來存取使用資料表來源的資料，然後使用資料表函數來轉換和篩選資料表資料。您可以使用 API 函數或 SQL 命令來轉換和篩選表格式資料。

本節包含下列主題：

- 資料表 API 連接器：這些元件可以在應用程式與外部資料來源和目的地之間移動資料。
- 資料表 API 時間屬性：本主題說明 Managed Service for Apache Flink 如何在使用資料表 API 時追蹤事件。

資料表 API 連接器

在 Apache Flink 程式設計模型中，連接器是應用程式用來從外部來源 (例如其他 AWS 服務) 讀取或寫入資料的元件。

透過 Apache Flink 資料表 API，您可以使用下列類型的連接器：

- 資料表 API 來源：您可以使用資料表 API 來源連接器以及 API 呼叫或 SQL 查詢，在 `TableEnvironment` 中建立資料表。
- 資料表 API 接收器：您可以使用 SQL 命令將資料表資料寫入外部來源，例如 Amazon MSK 主題或 Amazon S3 儲存貯體。

資料表 API 來源

您可以從資料串流建立資料表來源。下列程式碼會從 Amazon MSK 主題建立資料表：

```
//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);
```

如需關於資料表來源的詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的資料表和連接器。

資料表 API 接收器

若要將資料表資料寫入接收器，請在 SQL 中建立接收器，然後在 `StreamTableEnvironment` 物件上執行 SQL 型接收器。

下列程式碼範例示範如何將資料表資料寫入 Amazon S3 接收器：

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

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

您可以使用 `format` 參數來控制 Managed Service for Apache Flink 用來將輸出寫入接收器的格式。如需格式的相關資訊，請參閱《Apache Flink 文件》https://ci.apache.org/projects/flink/flink-docs-stable/中的格式。

如需關於資料表接收器的詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的資料表和連接器。

使用者定義的來源和接收器

您可以使用現有的 Apache Kafka 連接器與其他 AWS 服務 (例如 Amazon MSK 和 Amazon S3) 之間相互傳送資料。若要與其他資料來源和目的地互動，您可以定義自己的來源和接收器。如需詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的使用者定義的來源和接收器。

資料表 API 時間屬性

資料串流中的每個記錄都有數個時間戳記，用來定義與記錄相關的事件發生的時間：

- 事件時間：使用者定義的時間戳記，定義建立記錄的事件發生的時間。
- 擷取時間：應用程式從資料串流擷取記錄的時間。
- 處理時間：您的應用程式處理記錄的時間。

當 Apache Flink 資料表 API 根據記錄時間建立視窗時，您可以使用setStreamTimeCharacteristic 方法來定義 API 使用哪個時間戳記。

如需關於搭配資料表 API 使用時間戳記的詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的時間屬性。

搭配使用 Python 與 Managed Service for Apache Flink

> **Note**
>
> 如果您在使用 Apple 晶片的新 Mac 上開發 Python Flink 應用程式，可能會遇到與 PyfLink 1.15 的 Python 相依項的一些已知問題。在這種情況下，我們建議在 Docker 中執行 Python 解譯器。如需逐步說明，請參閱在 Apple Silicon Mac 上進行 PyfLink 1.15 開發。

Apache Flink 1.15.2 版包括使用 Python 版本 3.8 (使用 PyfLink 程式庫) 建立應用程式的支援。若要使用 Python 建立 Managed Service for Apache Flink 應用程式，請執行下列動作：

- 使用 `main` 方法將 Python 應用程式的程式碼建立為文字檔案。
- 將應用程式的程式碼檔案以及任何 Python 或 Java 相依性綁定到一個 zip 檔案中，然後將其上傳到 Amazon S3 儲存貯體。
- 建立 Managed Service for Apache Flink 應用程式，並指定 Amazon S3 程式碼位置、應用程式屬性和應用程式設定。

在高層級上，Python 資料表 API 是 Java 資料表 API 周圍的一項包裝函式。如需 Python 資料表 API 的相關資訊，請參閱《Apache Flink 文件》https://ci.apache.org/projects/flink/flink-docs-release-1.13/中的 Python 資料表 API 簡介。

編寫程式設計 Managed Service for Apache Flink Python 應用程式

您可以使用 Apache Flink Python 資料表 API 對 Managed Service for Apache Flink Python 應用程式進行編碼。Apache Flink 引擎將 Python 資料表 API 陳述式 (在 Python 虛擬機中執行) 轉換為 Java 資料表 API 陳述式 (在 Java 虛擬機中執行)。

執行下列動作以使用 Python 資料表 API：

- 建立對 `StreamTableEnvironment` 的參考。
- 透過對 `table` 參考執行查詢，從來源串流資料建立 `StreamTableEnvironment` 物件。
- 對 `table` 物件執行查詢以建立輸出資料表。
- 使用 `StatementSet` 將輸出資料表寫入目的地。

若要開始在 Managed Service for Apache Flink 中使用 Python 資料表 API，請參閱 適用於 Python 的 Amazon Managed Service for Apache Flink 入門。

讀取和寫入串流資料

若要讀取和寫入串流資料，請在資料表環境中執行 SQL 查詢。

建立資料表

下列程式碼範例示範可建立 SQL 查詢的使用者定義函數。SQL 查詢會建立與 Kinesis 串流互動的資料表：

```
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)
```

讀取串流資料

下列程式碼範例示範如何在資料表環境參考上使用先前的 CreateTable SQL 查詢來讀取資料：

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

寫入串流資料

下列程式碼範例示範如何使用 CreateTable 範例中的 SQL 查詢來建立輸出資料表參考，以及如何使用 StatementSet 與資料表互動，以將資料寫入目的地 Kinesis 串流：

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

讀取執行時間屬性

您可以使用執行時間屬性來設定應用程式，而無需變更應用程式的程式碼。

您可以指定應用程式的應用程式屬性，指定方式與 Managed Service for Apache Flink Java 應用程式相同。您可採用以下方式來指定執行時間屬性：

- 使用 CreateApplication 動作。
- 使用 UpdateApplication 動作。

- 使用主控台設定應用程式。

您可以讀取 Managed Service for Apache Flink 執行時間所建立的名為 `application_properties.json` 的 json 檔案，藉此擷取程式碼中的應用程式屬性。

下列程式碼範例示範從 `application_properties.json` 檔案讀取應用程式屬性：

```
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)
```

下列使用者定義函數的程式碼範例示範如何從應用程式屬性物件讀取屬性群組：擷取：

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

下列程式碼範例示範從上一個範例傳回的屬性群組讀取名為 INPUT_STREAM_KEY 的屬性：

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

建立應用程式的程式碼套件

建立 Python 應用程式之後，您就可以將程式碼檔案和相依性綁定到 zip 檔案中。

您的 zip 檔案必須包含帶有 `main` 方法的 Python 指令碼，並且可以選擇包含以下內容：

- 其他 Python 程式碼檔案
- JAR 檔案中使用者定義的 Java 程式碼
- JAR 檔案中的 Java 程式庫

> **Note**
>
> 應用程式 zip 檔案必須包含應用程式的所有相依性。您無法為應用程式參考其他來源的程式庫。

建立Managed Service for Apache Flink Python 應用程式

指定程式碼檔案

建立應用程式的程式碼套件之後，可將其上傳到 Amazon S3 儲存貯體。然後可以使用主控台或 CreateApplication 動作來建立應用程式。

使用 CreateApplication 動作建立應用程式時，可以使用名為 `kinesis.analytics.flink.run.options` 的特殊應用程式屬性群組在 zip 檔案中指定程式碼檔案和存檔。您可以定義下列類型的檔案：

- python：包含 Python 主要方法的文字檔案。
- jarfile：包含 Java 使用者定義函數的 Java JAR 檔案。
- pyFiles：包含應用程式要使用之資源的 Python 資源檔案。
- pyArchives：包含應用程式資源檔案的 zip 檔案。

如需關於 Apache Flink Python 程式碼檔案類型的詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的命令列用法。

> **Note**
>
> Managed Service for Apache Flink 不支援 `pyModule`、`pyExecutable`、或 `pyRequirements` 檔案類型。所有程式碼、請求和相依性都必須在 zip 檔案中。您無法指定要使用 pip 安裝的相依性。

以下範例 json 程式碼片段示範如何指定檔案在應用程式 zip 檔案中的位置：

```
"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"
          }
```

```
        },
```

監控 Managed Service for Apache Flink Python 應用程式

您可以使用應用程式的 CloudWatch 日誌來監控 Managed Service for Apache Flink Python 應用程式。

Managed Service for Apache Flink 記錄適用於 Python 應用程式的下列訊息：

- 在應用程式的 `main` 方法中使用 `print()` 寫入主控台的訊息。
- 使用 `logging` 套件以使用者定義函數的形式傳送的訊息。下列程式碼範例示範從使用者定義的函數寫入應用程式日誌：

```
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
```

- 應用程式擲出的錯誤訊息。

  如果應用程式在 `main` 函數中擲出例外狀況，該例外狀況將出現在應用程式的日誌中。

  下列範例示範從 Python 程式碼擲回例外狀況的日誌項目：

```
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**
>
> 由於效能問題，建議只在應用程式開發期間使用自訂日誌訊息。

使用 CloudWatch Insights 查詢日誌

下列 CloudWatch Insights 查詢會在執行應用程式的主函數時，搜尋 Python 進入點所建立的日誌：

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

Managed Service for Apache Flink 中的執行時間屬性

您可以使用執行時間屬性來設定應用程式，而無需重新編譯應用程式的程式碼。

本主題包含下列章節：

- 在主控台中使用執行時間屬性
- 在 CLI 中使用執行時間屬性
- 更新 Managed Service for Apache Flink 應用程式的執行時間屬性

在主控台中使用執行時間屬性

您可以使用主控台從 Managed Service for Apache Flink 新增、更新或移除執行時間屬性。

> **Note**
>
> 當您在 Managed Service for Apache Flink 主控台中建立應用程式時，無法新增執行時間屬性。

更新 Managed Service for Apache Flink 應用程式的執行時間屬性

1. 在以下網址開啟 Managed Service of Apache Flink 主控台：https://console.aws.amazon.com/flink
2. 選擇 Managed Service for Apache Flink 應用程式。選擇應用程式詳細資訊。
3. 在應用程式頁面中，選擇設定。
4. 展開屬性區段。
5. 使用屬性區段中的控制項，以索引鍵-值對定義屬性群組。可以使用這些控制項來新增、更新或移除屬性群組和執行時間屬性。
6. 選擇更新。

在 CLI 中使用執行時間屬性

您可以使用 AWS CLI 新增、更新或移除執行時間屬性。

本節包含針對設定應用程式執行時間屬性之 API 動作的範例請求。如需關於如何使用 JSON 檔案作為 API 動作輸入的資訊，請參閱 Managed Service for Apache Flink API 範例程式碼。

> **Note**
>
> 以您的帳戶 ID 取代以下範例中的範例帳戶 ID (*012345678901*)。

建立應用程式時新增執行時間屬性

`CreateApplication` 動作的下列範例請求會在您建立應用程式時新增兩個執行時間屬性群組 (`ProducerConfigProperties` 和 `ConsumerConfigProperties`)：

```
{
    "ApplicationName": "MyApplication",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-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"
               }
            }
         ]
      }
   }
}
```

在現有應用程式中新增和更新執行時間屬性

[UpdateApplication](#) 動作的下列範例請求會新增或更新現有應用程式的執行時間屬性：

```
{
  "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**
>
> 如果您使用的索引鍵在屬性群組中沒有對應的執行時間屬性，Managed Service for Apache Flink 會將索引鍵-值對新增為新屬性。如果您將索引鍵用於屬性群組中的現有執行時間屬性，Managed Service for Apache Flink 會更新屬性值。

移除執行時間屬性

UpdateApplication 動作的下列範例請求會從現有應用程式中移除所有執行時間屬性和屬性群組：

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

> **Important**
>
> 如果您省略現有屬性群組或屬性群組中的現有屬性索引鍵，則會移除該屬性群組或屬性。

更新 Managed Service for Apache Flink 應用程式的執行時間屬性

您可以使用可傳回 `Map<String, Properties>` 物件的靜態 `KinesisAnalyticsRuntime.getApplicationProperties()` 方法，在 Java 應用程式程式碼中擷取執行時間屬性。

下列 Java 程式碼範例會擷取應用程式的執行時間屬性：

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

您可以擷取屬性群組 (作為 `Java.Util.Properties` 物件)，如下所示：

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

您通常透過傳入 `Properties` 物件來設定 Apache Flink 來源或接收器，而無需擷取個別屬性。下列程式碼範例示範如何透過傳入從執行時間屬性擷取的 `Properties` 物件來建立 Flink 來源：

```
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;
}
```

如需使用執行時間屬性的完整程式碼範例，請參閱開始使DataStream 用 (API)。入門應用程式的來源程式碼可在 Managed Service for Apache Flink Java GitHub 儲存庫的入門中取得。

在 Managed Service for Apache Flink 中實作容錯能力

檢查點是用於在 Amazon Managed Service for Apache Flink 中實作容錯能力的方法。檢查點是執行中應用程式的最新備份，可用來從意外的應用程式中斷或容錯移轉中立即復原。

如需 Apache Flink 應用程式中檢查點的詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的檢查點。

快照是手動建立和管理的應用程式狀態備份。快照可讓您透過呼叫 UpdateApplication 將應用程式還原到先前的狀態。如需詳細資訊，請參閱使用快照管理應用程式備份。

如果應用程式已啟用檢查點，則服務會在意外的應用程式重新啟動時建立並載入應用程式資料的備份，藉此提供容錯能力。這些意外的應用程式重新啟動可能是由於意外的作業重新啟動、執行個體失敗等原因造成。這為應用程式提供了與在這些重新啟動期間無故障執行時相同的語義。

如果已為應用程式啟用快照，並使用應用程式的 Application RestoreConfiguration 進行設定，則服務會在應用程式更新期間或在與服務相關的擴展或維護期間提供恰好一次的處理語義。

在 Managed Service for Apache Flink 設定檢查點

您可以設定應用程式的檢查點行為。您可以定義應用程式是否保持檢查點狀態、應用程式將狀態儲存至檢查點的頻率，以及某個檢查點操作結束與另一個檢查點操作開始之間的最小間隔。

您可以使用 CreateApplication 或 UpdateApplication API 作業來設定下列設定：

- CheckpointingEnabled：指示是否已在應用程式中啟用檢查點。
- CheckpointInterval：包含檢查點 (持續性) 作業之間的時間 (毫秒)。
- ConfigurationType：將此值設定為 DEFAULT 以使用預設檢查點行為。將此值設定為 CUSTOM 以設定其他值。

> **Note**
>
> 預設的檢查點行為如下：
>
> - CheckpointingEnabled：true
> - CheckpointInterval：60000
> - MinPauseBetweenCheckpoints：5000
>
> 如果 ConfigurationType 設為 DEFAULT，即使前述值使用 AWS Command Line Interface 或在應用程式程式碼中設為其他值，仍會使用前述值。

> **Note**
>
> 對於 Flink 1.15 以上版本，Managed Service for Apache Flink 將在自動建立快照期間 (也就是應用程式更新、擴展或停止時) 使用 `stop-with-savepoint`。

- `MinPauseBetweenCheckpoints`：檢查點操作結束與另一個檢查點操作開始之間的最短時間 (毫秒)。設定此值可防止在檢查點操作時間超過 `CheckpointInterval` 時，應用程式持續執行檢查點。

檢查點 API 範例

本節包含針對設定應用程式檢查點之 API 動作的範例請求。如需關於如何使用 JSON 檔案作為 API 動作輸入的資訊，請參閱 Managed Service for Apache Flink API 範例程式碼。

設定新應用程式的檢查點

`CreateApplication` 動作的下列請求範例會在您建立應用程式時設定檢查點：

```
{
   "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
         }
      }
```

```
}
```

停用新應用程式的檢查點

CreateApplication 動作的下列請求範例會在您建立應用程式時停用檢查點：

```
{
   "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"
         }
      }
}
```

設定現有應用程式的檢查點

UpdateApplication 動作的下列範例請求會為現有的應用程式設定檢查點：

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

```
}
```

停用現有應用程式的檢查點

UpdateApplication 動作的下列範例請求會為現有的應用程式停用檢查點：

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

使用快照管理應用程式備份

快照是 Apache Flink 儲存點的 Managed Service for Apache Flink 實作。快照是使用者或服務觸發、建立和管理的應用程式狀態備份。如需關於 Apache Flink 儲存點的資訊，請參閱《Apache Flink 文件》中的儲存點。https://nightlies.apache.org/flink/flink-docs-release-1.15/使用快照，您可以從應用程式狀態的特定快照重新啟動應用程式。

> **Note**
>
> 建議您的應用程式每天建立數次快照，以便使用正確的狀態資料正確重新啟動。快照的正確頻率取決於應用程式的業務邏輯。頻繁拍攝快照可讓您復原較新的資料，但會增加成本並需要更多的系統資源。

在 Managed Service for Apache Flink 中，您使用下列 API 動作來管理快照：

- CreateApplicationSnapshot
- DeleteApplicationSnapshot
- DescribeApplicationSnapshot

- ListApplicationSnapshots

如需每個應用程式的快照數目限制，請參閱配額。如果您的應用程式達到快照數目限制，則手動建立快照會失敗，並顯示LimitExceededException。

Managed Service for Apache Flink 永遠不會刪除快照。您必須使用 DeleteApplicationSnapshot 動作手動刪除快照。

若要在啟動應用程式時載入儲存的應用程式狀態快照，請使用 StartApplication 或 UpdateApplication 動作的 ApplicationRestoreConfiguration 參數。

本主題包含下列章節：

- 自動建立快照
- 從包含不相容狀態資料的快照還原
- 快照 API 範例

自動建立快照

如果SnapshotsEnabled 在應用程式的 ApplicationSnapshotConfiguration 中設定為 true，Managed Service for Apache Flink 會在應用程式更新、擴展或停止時自動建立並使用快照，以提供恰好一次的處理語義。

> **Note**
>
> 將 ApplicationSnapshotConfiguration::SnapshotsEnabled 設定為 false 會導致應用程式更新期間資料遺失。

> **Note**
>
> Managed Service for Apache Flink 會在快照建立期間觸發中繼儲存點。對於 Flink 版本 1.15 或更高版本，中繼儲存點不會再造成任何副作用。請參閱觸發儲存點

自動建立的快照具有下列特質：

- 快照由服務管理，但您可以使用 ListApplicationSnapshots 動作查看快照。自動建立的快照會根據您的快照限制計數。

- 如果您的應用程式超過快照限制，手動建立的快照將會失敗，但 Managed Service for Apache Flink 服務在應用程式更新、擴展或停止時仍會成功建立快照。您必須先使用 DeleteApplicationSnapshot 動作手動刪除快照，然後才能手動建立更多快照。

從包含不相容狀態資料的快照還原

由於快照包含運算子的資訊，因此從快照還原運算子的狀態資料 (自上一應用程式版本以來已變更) 可能會產生非預期的結果。如果應用程式嘗試從不對應於目前運算子的快照還原狀態資料，應用程式將會發生錯誤。錯誤的應用程式將卡在 `STOPPING` 或 `UPDATING` 狀態。

若要允許應用程式從包含不相容狀態資料的快照還原，請使用 UpdateApplication 動作將 FlinkRunConfiguration 的 `AllowNonRestoredState` 參數設定為 `true`。

從過時的快照還原應用程式時，您會看到下列行為：

- 新增運算子：如果新增了新的運算子，則儲存點沒有該新運算子的狀態資料。不會發生任何錯誤，並且不必設定 `AllowNonRestoredState`。
- 刪除運算子：如果現有的運算子被刪除，則儲存點會有該遺失運算子的狀態資料。除非將 `AllowNonRestoredState` 設定為 `true`，否則會發生錯誤。
- 修改運算子：如果進行了相容的變更，例如將參數類型變更為相容類型，應用程式就可以從過時的快照還原。如需關於從快照還原的詳細資訊，請參閱《Apache Flink 文件》中的儲存點。使用 Apache Flink 1.8 版或更新版本的應用程式可能可以從具有不同結構描述的快照還原。使用 Apache Flink 1.6 版的應用程式無法還原。對於兩階段遞交接收器，建議使用系統快照 (SwS)，而不是使用者建立的快照 (CreateApplicationSnapshot)。

  對於 Flink，Managed Service for Apache Flink 會在快照建立期間觸發中繼儲存點。對於 Flink 版本 1.15 以上版本，中繼儲存點不會再造成任何副作用。請參閱觸發儲存點。

如果您需要恢復與現有儲存點資料不相容的應用程式，建議將 StartApplication 動作的 `ApplicationRestoreType` 參數設定為 `SKIP_RESTORE_FROM_SNAPSHOT`，以略過從快照還原。

如需關於 Apache Flink 如何處理不相容狀態資料的詳細資訊，請參閱《Apache Flink 文件》中的狀態結構描述演進。

快照 API 範例

本節包括將快照與應用程式搭配使用的 API 動作的範例請求。如需關於如何使用 JSON 檔案作為 API 動作輸入的資訊，請參閱 Managed Service for Apache Flink API 範例程式碼。

列出應用程式的快照

UpdateApplication 動作的下列請求範例可為應用程式啟用快照：

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

建立快照

CreateApplicationSnapshot 動作的下列範例請求可建立目前應用程式狀態的快照：

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

列出應用程式的快照

ListApplicationSnapshots 動作的下列範例請求會列出目前應用程式狀態的前 50 個快照：

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

列出應用程式的快照詳細資訊

DescribeApplicationSnapshot 動作的下列請求範例會列出特定應用程式快照的詳細資訊：

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

刪除快照

DeleteApplicationSnapshot 動作的下列請求範例會刪除先前儲存的快照。您可以使用 ListApplicationSnapshots或 DeleteApplicationSnapshot 取得 SnapshotCreationTimestamp 值：

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

使用具名快照重新啟動應用程式

StartApplication 動作的下列請求範例會使用特定快照中的已儲存狀態來啟動應用程式：

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

使用最新的快照重新啟動應用程式

StartApplication 動作的下列請求範例會使用最新快照來啟動應用程式：

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

不使用快照重新啟動應用程式

`StartApplication` 動作的下列請求範例會在不載入應用程式狀態的情況下啟動應用程式，即使有快照也是如此：

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

在 Managed Service for Apache Flink 中進行應用程式擴展

您可以為 Amazon Managed Service for Apache Flink 設定任務的平行執行和資源配置，以實作擴展。如需 Apache Flink 排程任務平行執行個體的相關資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的平行執行。

主題

- 設定應用程式平行程度和 KPU ParallelismPer
- 配置 Kinesis 處理單元
- 更新應用程式的平行處理
- 自動擴展

設定應用程式平行程度和 KPU ParallelismPer

您可以使用下列 `ParallelismConfiguration` 屬性，為 Managed Service for Apache Flink 應用程式任務 (例如從來源讀取或執行運算子) 設定平行執行：

- `Parallelism`：使用此屬性可設定預設的 Apache Flink 應用程式平行處理層級。除非在應用程式程式碼中覆寫，否則所有運算子、來源和接收器都按此平行處理層級執行。預設值為 1，最大值為 256。
- `ParallelismPerKPU`：使用此屬性設定依應用程式每 Kinesis 處理單元 (KPU) 可排程的平行任務數目。預設值為 1，最大值為 8。對於具有封鎖作業 (例如 I/O) 的應用程式，較高的 `ParallelismPerKPU` 值會導致 KPU 資源的完整使用率。

> **Note**
>
> `Parallelism` 的限制等於 KPU 的限制 `ParallelismPerKPU` 乘以 (預設值為 64)。KPU 限制可透過請求提高限制來增加。如需如何請求提高限制的指示，請參閱 Service Quotas 中的「請求提高限制」。

如需為特定運算子設定任務平行處理層級的相關資訊，請參閱《Apache Flink 說明文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的設定平行處理層級：運算子。

配置 Kinesis 處理單元

Managed Service for Apache Flink 以 KPU 的形式佈建容量。單一 KPU 可為您提供 1 個 vCPU 和 4 GB 的記憶體。針對每個配置的 KPU，還會提供 50 GB 的執行中應用程式儲存體。

Managed Service for Apache Flink 會使用 `Parallelism` 和 `ParallelismPerKPU` 屬性計算執行應用程式所需的 KPU，如下所示：

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

Managed Service for Apache Flink 可快速提供應用程式資源，以因應輸送量或處理活動尖峰。它會在活動尖峰過去後逐漸從應用程式中移除資源。若要停用資源的自動配置，請將 `AutoScalingEnabled` 值設定為 `false`，如稍後 更新應用程式的平行處理 中所述。

應用程式的 KPU 預設限制為 64。如需如何請求提高此限制的指示，請參閱 Service Quotas 中的「請求提高限制」。

> **Note**
>
> 額外的 KPU 需要為了協同運作目的付費。如需詳細資訊，請參閱 Managed Service for Apache Flink 定價。

更新應用程式的平行處理

本節包含設定應用程式平行處理之 API 動作的範例請求。如需關於如何將請求區塊與 API 動作搭配使用的更多範例和指示，請參閱Managed Service for Apache Flink API 範例程式碼。

CreateApplication 動作的下列請求範例會在您建立應用程式時設定平行處理：

```
{
   "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 動作的下列請求範例會為現有的應用程式時設定平行處理：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "ParallelismConfigurationUpdate": {
            "AutoScalingEnabledUpdate": "true",
            "ConfigurationTypeUpdate": "CUSTOM",
            "ParallelismPerKPUUpdate": 4,
            "ParallelismUpdate": 4
         }
      }
   }
```

```
}
```

UpdateApplication 動作的下列請求範例會為現有的應用程式時停用平行處理：

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

自動擴展

Managed Service for Apache Flink 可彈性擴展應用程式的平行處理層級，以因應來源的資料輸送量和運算子在大多數情況下的複雜性。Managed Service for Apache Flink 會監控應用程式的資源 (CPU) 使用情況，並相應地彈性擴展應用程式的平行處理層級：

- 如果指標大於 75% 或更高，持續 15 分鐘，您containerCPUUtilization的應用程式可擴展 (增加平行 CloudWatch 度)。這表示當存在 15 個連續資料點且 1 分鐘期間等於或超過 75% 時，就會觸發 ScaleUp 動作。
- 當 CPU 使用率維持在 10% 以下達 6小時時，應用程式會縮減規模 (減少平行處理層級)。這表示當存在 360 個連續資料點且 1 分鐘期間少於 10% 時，就會觸發 ScaleDown 動作。

> **Note**
>
> 可以參考 1 分鐘時間段內 containerCPUUtilization 的最大值，以找出與用於擴展動作之資料點的關聯性，但不需要反映動作觸發時的確切時刻。

Managed Service for Apache Flink 不會將應用程式的 CurrentParallelism 值降低到小於應用程式的 Parallelism 設定。

當 Managed Service for Apache Flink 服務擴展應用程式時，應用程式將處於 AUTOSCALING 狀態。您可以使用 DescribeApplication或 ListApplications動作來檢查目前的應用程式狀態。當服務擴展您的應用程式時，唯一可以使用的有效 API 動作是 StopApplication將Force參數設定為true。

您可以使用 AutoScalingEnabled 屬性 (FlinkApplicationConfiguration 的一部分) 來啟用或停用自動擴展行為。您的 AWS 帳戶需支付 Managed Service for Apache Flink 佈建的 KPU 費用，這是您應用程式的 parallelism 和 parallelismPerKPU 設定的功能。活動尖峰會增加您的 Managed Service for Apache Flink 成本。

如需定價相關的資訊，請參閱 Amazon Managed Service for Apache Flink 定價。

請留意下列與應用程式擴展相關的資訊：

- 預設會啟用自動擴展。
- 擴展不適用於 Studio 筆記本。不過，如果您將 Studio 筆記本部署為具有持久狀態的應用程式，則擴展將套用到已部署的應用程式。
- 應用程式的預設限制為 64 個 KPU。如需詳細資訊，請參閱 配額。
- 當自動擴展更新應用程式的平行處理層級時，應用程式會經歷停機。若要避免停機，請執行下列動作：
  - 停用自動擴展
  - 配置您的應用程序parallelism和parallelismPerKPUUpdateApplication操作。如需設定應用程式平行處理設定的詳細資訊，請參閱下文的the section called “更新應用程式的平行處理”。
  - 定期監控應用程式的資源使用量，以確認應用程式的工作負載具有正確的平行處理設定。如需資源配置情況的相關資訊，請參閱the section called “Managed Service for Apache Flink 中的指標和維度”。

maxParallelism 考量

- 自動擴展邏輯可防止將 Flink 作業擴展至會對作業和 maxParallelism 運算子造成干擾的平行處理層級。例如，如果一個簡單的作業只有一個來源和一個接收器，其中來源 maxParallelism 為 16，sink 為 8，則我們不會將作業自動擴展到 8 以上。
- 如果未針對作業設定 maxParallelism，Flink 會將其預設為 128。因此，如果您認為作業需要以高於 128 的平行處理層級執行，則必須為應用程式設定該數字。
- 如果您應該看到作業自動擴展，但沒有看到它，請確保您的 maxParallelism 值允許。

如需更多資訊，請參閱 Apache Flink 的增強型監控和自動擴展

如需範例，請參閱 kda-flink-app-autoscaling。

使用標籤

本節說明如何將索引鍵-值中繼資料標籤新增至 Managed Service for Apache Flink 應用程式。這些標籤可用於下列目的：

- 決定個別 Managed Service for Apache Flink 應用程式的帳單。如需詳細資訊，請參閱《帳單和成本管理使用者指南》中的使用成本分配標籤。
- 根據標籤控制對應用程式資源的存取。如需詳細資訊，請參閱《AWS Identity and Access Management IAM 使用者指南》中的使用標籤控制存取權。
- 使用者定義的目的。您可以根據使用者標籤的存在來定義應用程式的功能。

注意有關標籤的以下資訊：

- 應用程式標籤的數目上限包括系統標籤。使用者定義的應用程式的標籤數目上限為 50。
- 如果動作包含具有重複 `Key` 值的標籤清單，則服務會擲出 `InvalidArgumentException`。

本主題包含下列章節：

- 建立應用程式時新增標籤
- 為現有應用程式新增或更新標籤
- 列出應用程式的標籤
- 從應用程式移除標籤

建立應用程式時新增標籤

您可以在使用 CreateApplication 動作的 `tags` 參數建立應用程式時新增標籤。

以下範例請求顯示 `CreateApplication` 請求的 `Tags` 節點：

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

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

為現有應用程式新增或更新標籤

您可以使用 TagResource 動作將標籤新增至應用程式。您無法使用 UpdateApplication 動作為應用程式新增標籤。

若要更新現有的標籤，請使用與現有標籤相同的索引鍵新增標籤。

TagResource 動作的下列請求範例會新增標籤或更新現有標籤：

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

列出應用程式的標籤

若要列出現有標籤，請使用 ListTagsForResource 動作。

ListTagsForResource 動作的下列請求範例可列出應用程式的標籤：

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

從應用程式移除標籤

若要從應用程式移除標籤，請使用 UntagResource 動作。

UntagResource 動作的下列請求範例可移除應用程式的標籤：

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

搭配使用 CloudFormation 與 Managed Service for Apache Flink

下列練習說明如何使用相同堆疊中的 Lambda 函數啟動透過 AWS CloudFormation 建立的 Flink 應用程式。

開始之前

開始本練習之前，請按照 AWS::KinesisAnalytics::Application 中的步驟，使用 AWS CloudFormation 建立 Flink 應用程式。

編寫 Lambda 函數

在建立或更新 Flink 應用程式後，若要啟動它，可以使用kinesisanalyticsv2 start-application API。建立 Flink 應用程式後，呼叫將由 AWS CloudFormation 事件觸發。在本練習稍後部分，我們將討論如何設定堆疊以觸發 Lambda 函數，但我們先專注於 Lambda 函數宣告及其程式碼。我們在本範例中使用 Python3.8 運行時間。

```
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)})
```

在前面的程式碼中，Lambda 將處理傳入的 AWS CloudFormation 事件、篩除 `Create` 和 `Update` 以外的所有內容、並獲取應用程式狀態並在狀態為 READY 時啟動它。為了獲取應用程式狀態，您需要建立 Lambda 角色，如下所示：

建立 Lambda 角色

您可以為 Lambda 建立角色，以便與應用程式成功「通話」並寫入日誌。此角色將使用預設的受管政策，但您可以使用自訂政策來縮小其範圍。

```
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/Amazonmanaged-flinkFullAccess
        - arn:aws:iam::aws:policy/CloudWatchLogsFullAccess
      Path: /
```

請注意，Lambda 資源將在建立 Flink 應用程式之後在同一堆疊中建立，因為它們依賴於它。

調用 Lambda 函數

現在剩下要做的就是調用 Lambda 函數。這使用自訂資源來完成。

```
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
```

以上是使用 Lambda 啟動 Flink 應用程式所需的一切。您現在可以建立自己的堆疊，也可以使用下面的完整範例來查看所有這些步驟的實際運作方式。

完整範例

以下範例是上述步驟的稍微擴展版本，透過模板參數進行了額外的 RunConfiguration 調整。這是一個工作堆疊供您嘗試。請務必閱讀隨附的注意事項：

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:
                - kinesisanlaytics.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/Amazonmanaged-flinkFullAccess
        - 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
```

您也可以調整 Lambda 的角色以及應用程式本身的角色。

在建立上面的堆疊之前，不要忘記指定參數。

parameters.json

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

以您的特定需求取代 YOUR_BUCKET_ARN 和 YOUR_JAR。您可以按照本指南來建立 Amazon S3 儲存貯體和應用程式 jar。

現在建立堆疊 (以您選擇的區域，例如 US-east-1，取代 YOUR_REGION)：

```
aws cloudformation create-stack --region YOUR_REGION --template-body "file://
stack.yaml" --parameters "file://parameters.json" --stack-name "TestManaged Service for
 Apache FlinkStack" --capabilities CAPABILITY_NAMED_IAM
```

現在，您可以導航到 https://console.aws.amazon.com/cloudformation 並檢視進度。建立 Flink 應用程式後，您應該會看到該應用程式處於 `Starting` 狀態。可能需要幾分鐘的時間才開始 `Running`。

如需詳細資訊，請參閱下列內容：

- 使用 AWS CloudFormation (第 1 部分，共 3 部分) 擷取任何 AWS 服務屬性的四種方法。
- 逐步導覽：查詢 Amazon Machine Image ID。

將 Apache Flink 儀表板與 Managed Service for Apache Flink 搭配使用

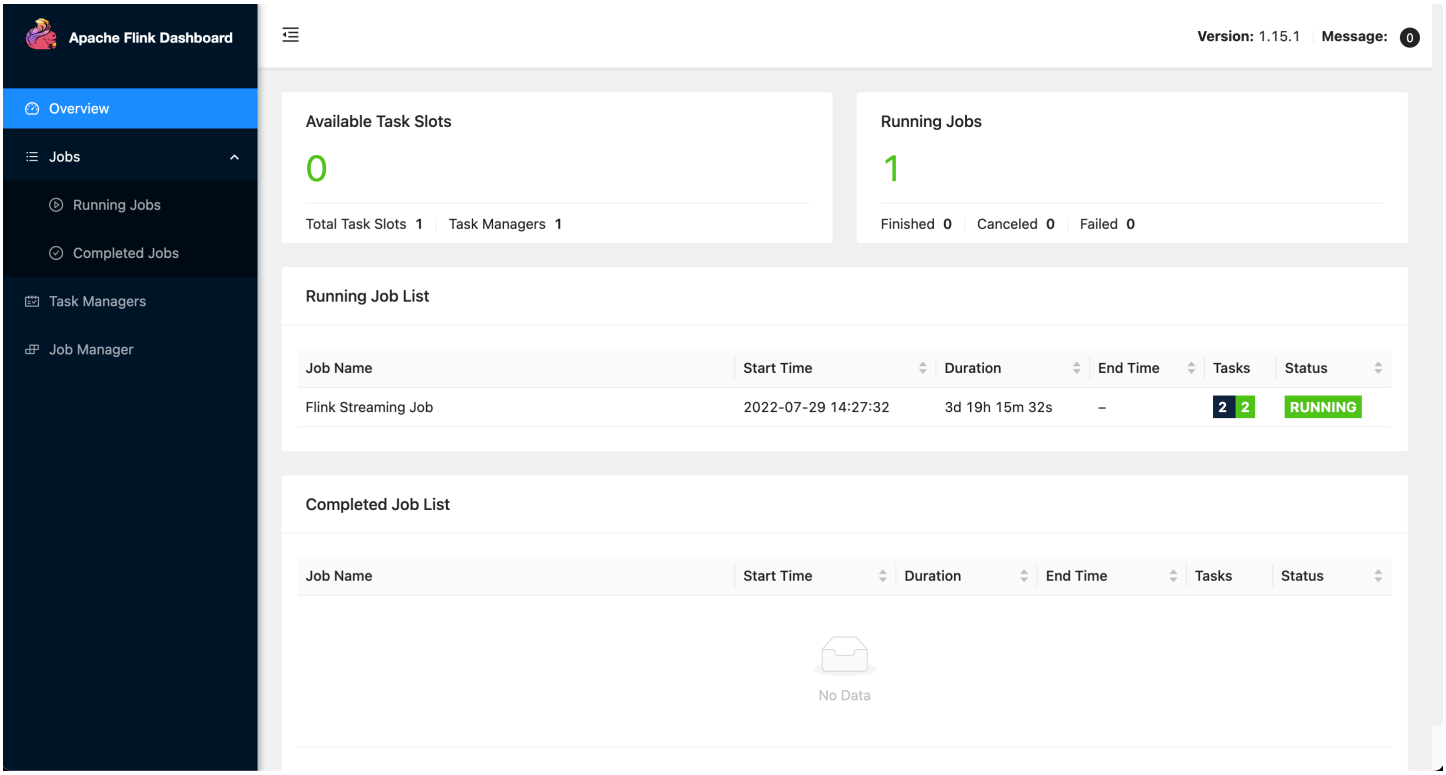
您可以使用應用程式的 Apache Flink 儀表板來監控 Managed Service for Apache Flink 應用程式的運作狀態。應用程式的儀表板顯示下列資訊：

- 使用中的資源，包括任務管理員和任務插槽。
- 作業資訊，包括正在執行、已完成、已取消和失敗的作業。

如需關於 Apache Flink 任務管理員、任務插槽和作業的詳細資訊，請參閱 Apache Flink 網站上的 Apache Flink 架構。

請注意下列將 Apache Flink 儀表板用於 Managed Service for Apache Flink 應用程式的相關事項：

- 用於 Managed Service for Apache Flink 應用程式的 Apache Flink 儀表板是唯讀的。您無法使用 Apache Flink 儀表板變更 Managed Service for Apache Flink 應用程式。
- Apache Flink 儀表板與 Microsoft Internet Explorer 不相容。



存取應用程式的 Apache Flink 儀表板

您可以透過 Managed Service for Apache Flink 主控台存取應用程式的 Apache Flink 儀表板，也可以透過使用 CLI 請求安全 URL 端點。

使用 Managed Service for Apache Flink 主控台存取應用程式的 Apache Flink 儀表板

若要從主控台存取應用程式的 Apache Flink 儀表板，請在應用程式頁面上選擇 Apache Flink 儀表板。

> **Note**
>
> 從 Managed Service for Apache Flink 主控台開啟儀表板時，主控台會產生 URL，其有效時間為 12 小時。

使用 Managed Service for Apache Flink CLI 存取應用程式的 Apache Flink 儀表板

您可以使用 Managed Service for Apache Flink CLI 產生 URL 來存取應用程式儀表板。所產生的 URL 在指定的時間內有效。

> **Note**
>
> 如果您在三分鐘內未存取產生的 URL，則該 URL 將不再有效。

您可以使用 CreateApplicationPresignedUrl 動作來產生儀表板 URL。您可以為動作指定下列參數值：

- 應用程式名稱
- URL 有效時間 (秒)
- 您可以指定 `FLINK_DASHBOARD_URL` 為 URL 類型。

發行版本

本主題包含每個 Managed Service for Apache Flink 發行版本所支援的功能和建議元件版本的相關資訊。

Amazon Managed Service for Apache Flink 1.15.2 發行版

Managed Service for Apache Flink 支援 Apache 1.15.2 以下新功能

功能	說明	Apache FLIP 參考
非同步接收器	AWS 貢獻的用於建置非同步目的地的架構，可讓開發人員以不到以前一半的努力來建置自訂 AWS 連接器。如需詳細資訊，請參閱通用非同步基本接收器。	FLIP-171：非同步接收器。
Kinesis Data Firehose 接收器	AWS 使用非同步框架貢獻了一個新的 Amazon Kinesis Firehose 接收器。	Amazon Kinesis Data Firehose 接收器。
使用儲存點停止	「使用儲存點停止」可確保乾淨利落的停止操作，最重要的是為依賴它們的客戶提供了僅支援一次的語義。	FLIP-34：使用儲存點終止/暫停作業。
Scala 解耦	使用者現在可以利用任何 Scala 版本的 Java API，包括 Scala 3。客戶需要將所選擇的 Scala 標準程式庫綁定在他們的 Scala 應用程式中。	FLIP-28：將移除 flink-table 的 Scala 相依性作為長期目標。
Scala	請參閱上面的 Scala 解耦	FLIP-28：將移除 flink-table 的 Scala 相依性作為長期目標。

功能	說明	Apache FLIP 參考
統一的連接器指標	Flink 針對作業、任務和運算子擁有已定義的標準指標。Managed Service for Apache Flink 將繼續支援接收器和來源指標，並在 1.15 版中為可用性指標同時引入了 `numRestarts` 與 `fullRestarts` 。	FLIP-33：將連接器指標標準化和 FLIP-179：公開標準化的運算子指標。
檢查點已完成的任務	此功能在 Flink 1.15 中預設為啟用，即使作業圖表的某些部分已完成處理所有資料 (如果包含綁定的 (批次) 來源，可能會發生此情況)，仍可以繼續執行檢查點。	FLIP-147：在任務完成後支援檢查點。

使用 Apache Flink 1.15 的 Amazon Managed Service for Apache Flink 中的變更

Studio 筆記本

Managed Service for Apache Flink Studio 現支援 Apache Flink 1.15。Managed Service for Apache Flink Studio 利用 Apache Zeppelin 筆記本提供單一介面開發體驗，用於開發、程式碼偵錯和執行 Apache Flink 串流處理應用程式。您可以在 將 Studio 筆記本用於 Managed Service for Apache Flink 中進一步了解 Managed Service for Apache Flink Studio 以及如何開始使用。

EFO 連接器

升級至 Managed Service for Apache Flink 1.15 版時，確保使用的是最新的 EFO 連接器，也就是任何 1.15.3 版或更新版本。如需關於原因的詳細資訊，請參閱 FLINK-29324。

Scala 解耦

從 Flink 1.15.2 開始，您需要將您選擇的 Scala 標準程式庫綁定到 Scala 應用程式中。

Kinesis Data Firehose 接收器

升級至 Managed Service for Apache Flink 1.15 版時，確保使用的是最新的 Amazon Kinesis Data Firehose 接收器。

Kafka 連接器

升級至 Amazon Managed Service for Apache Flink 1.15 版時，確保使用的是最新的 Kafka 連接器 API。Apache Flink 已不推薦使用 FlinkKafkaConsumer 和 FlinkKafkaProducer。對於 Flink 1.15，這些用於 Kafka 接收器的 API 無法遞交給 Kafka。確保您正在使用 KafkaSource 和 KafkaSink。

元件

元件	版本
Java	11 (建議使用)
Scala	2.12
Managed Service for Apache Flink 執行時間 (aws-kinesisanalytics-runtime)	1.2.0
AWS Kinesis 連接器 (flink-connector-kinesis)	1.15.4
Apache Beam (僅限於 Beam 應用程式)	2.33.0，帶有 Jackson 2.12.2 版

將 Studio 筆記本用於 Managed Service for Apache Flink

適用於 Managed Service for Apache Flink 的 Studio 筆記本可讓您即時以互動方式查詢資料串流，並使用標準 SQL、Python 和 Scala 輕鬆建置和執行串流處理應用程式。只要在 AWS 管理主控台按幾下滑鼠，即可啟動無伺服器筆記本，以查詢資料串流並在幾秒鐘內取得結果。

筆記本是基於 Web 的開發環境。使用筆記本，您不僅能獲得簡單的互動式開發體驗，還能使用 Apache Flink 提供的進階功能。Studio 筆記本使用由 Apache Zeppelin 提供支援的筆記本，使用 Apache Flink 作為串流處理引擎。Studio 筆記本無縫結合了這些技術，讓所有技能背景的開發人員都能存取資料串流的進階分析。

Apache Zeppelin 為您的 Studio 筆記本提供了完整的分析工具套件，包括以下專案：

- 資料視覺化
- 將資料匯出到檔案
- 控制輸出格式以便於分析

若要開始使用 Managed Service for Apache Flink 和 Apache Zeppelin，請參閱建立 Studio 筆記本教學課程。如需關於 Apache Zeppelin 的詳細資訊，請參閱《Apache Zeppelin 文件》http://zeppelin.apache.org。

使用筆記本，您可以使用 SQL，Python 或斯卡拉中的 Apache Flink 表 API 和 SQL 或斯卡拉或斯卡拉中的 DataStream API 來建模查詢。只需點按幾下，即可將 Studio 筆記本升級為適用於生產工作負載的、持續執行的、非互動式 Managed Service for Apache Flink 串流處理應用程式。

本主題包含下列章節：

- 建立 Studio 筆記本
- 串流資料的互動式分析
- 部署為具有持久狀態的應用程式
- Studio 筆記本的 IAM 許可
- 連接器和相依性
- 使用者定義的函數
- 啟用檢查點
- 使用 AWS Glue

- 範例與教學課程
- 故障診斷
- 附錄：建立自訂 IAM 政策

建立 Studio 筆記本

Studio 筆記本包含用 SQL、Python 或 Scala 編寫的查詢或程式，這些查詢或程式可以在串流資料上執行並傳回分析結果。您使用主控台或 CLI 建立應用程式，並提供查詢以分析資料來源中的資料。

應用程式具有下列元件：

- 資料來源，例如 Amazon MSK 叢集、Kinesis 資料串流或 Amazon S3 儲存貯體。
- AWS Glue 資料庫。此資料庫包含儲存資料來源、目標結構描述和端點的資料表。如需詳細資訊，請參閱 使用 AWS Glue。
- 應用程式的程式碼。您的程式碼會實作您的分析查詢或程式。
- 您的應用程式設定和執行時間屬性。如需應用程式設定和執行時間屬性的相關資訊，請參閱《Apache Flink 應用程式開發人員指南》https://docs.aws.amazon.com/managed-flink/latest/java/what-is.html中的下列主題：
  - 應用程式平行處理和擴展：您可以使用應用程式的平行處理設定來控制應用程式可同時執行的查詢數目。如果查詢具有多個執行路徑，則還可以利用增加的平行處理，如下列情況所示：
    - 處理 Kinesis 資料串流的多個碎片時
    - 使用 KeyBy 運算子分割資料時。
    - 使用多個視窗運算子時

    如需關於應用程式擴展的詳細資訊，請參閱 Managed Service for Apache Flink 中的應用程式擴展。
  - 記錄和監控：如需應用程式記錄和監控的相關資訊，請參閱 Amazon Managed Service for Apache Flink 中的記錄和監控。
  - 應用程式使用檢查點和儲存點進行容錯。依預設，Studio 筆記本不啟用檢查點和儲存點。

您可以使用 AWS Management Console 或 AWS CLI 建立 Studio 筆記本。

從主控台建立應用程式時，您可以使用下列選項：

- 在 Amazon MSK 主控台中，選擇您的叢集，然後選擇即時處理資料。

- 在 Kinesis Data Streams 主控台中，選擇您的資料串流，然後在應用程式標籤上選擇即時處理資料。
- 在 Managed Service for Apache Flink 主控台中，選擇 Studio 標籤，然後選擇建立 Studio 筆記本。

如需教學課程，請參閱使用 Managed Service for Apache Flink 進行事件偵測。

如需更進階 Studio 筆記本解決方案的範例，請參閱 Apache Flink 用於 Amazon Managed Service for Apache Flink Studio。

串流資料的互動式分析

您使用由 Apache Zeppelin 提供支援的無伺服器筆記本與串流資料互動。您的筆記本可以包含多條筆記，每條筆記可以有一個或多個段落，可以在其中撰寫程式碼。

下列範例 SQL 查詢顯示如何從資料來源擷取資料：

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

如需 Flink 串流 SQL 查詢的更多範例，請參閱下文的 範例與教學課程，以及《Apache Flink 文件》中的查詢https://nightlies.apache.org/flink/flink-docs-release-1.15/。

您可以在 Studio 筆記本中使用 Flink SQL 查詢來查詢串流資料。也可以使用 Python (資料表 API) 和 Scala (資料表和 Datastream API) 編寫程式，以互動方式查詢串流資料。您可以檢視查詢或程式的結果，在幾秒鐘內更新它們，然後重執行以檢視更新的結果。

Flink 解譯器

您可以使用解譯器指定 Managed Service for Apache Flink 用來執行應用程式的語言。您可以將下列解譯器用於 Managed Service for Apache Flink：

名稱	類別	描述
%flink	FlinkInterpreter	Creates ExecutionEnvironment/StreamExecutionEnvironment/BatchTableEnvironment/StreamTableEnvironment and provides a Scala environment

名稱	類別	描述
%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

如需 Flink 解譯器的詳細資訊，請參閱 Apache Zeppelin 的 Flink 解譯器。

如果您使用 `%flink.pyflink` 或 `%flink.ipyflink` 作為解譯器，則需要使用 `ZeppelinContext` 來視覺化筆記本內的結果。

如需更 PyFlink 具體的範例，請參閱使用 Apache Flink 工作室和 Python 的受管理服務以互動方式查詢您的資料串流。

Apache Flink 資料表環境變數

Apache Zeppelin 使用環境變數提供對資料表環境資源的存取。

您可以使用以下變數存取 Scala 資料表環境資源：

變數	資源
`senv`	`StreamExecutionEnvironment`
`stenv`	`StreamTableEnvironment ######`

您可以使用以下變數存取 Python 資料表環境資源：

變數	資源
`s_env`	`StreamExecutionEnvironment`
`st_env`	`StreamTableEnvironment ######`

如需有關使用資料表環境的詳細資訊，請參閱 TableEnvironment在 Apache Flink 文件中建立一個。

部署為具有持久狀態的應用程式

您可以建立程式碼並將其匯出到 Amazon S3。您可以將在筆記中撰寫的程式碼升級為持續執行的串流處理應用程式。在 Managed Service for Apache Flink 上執行 Apache Flink 應用程式有兩種模式：使用 Studio 筆記本，您可以互動方式開發程式碼、即時檢視程式碼結果，並在筆記中以視覺化方式呈現。您將筆記部署為在串流模式下執行後，Managed Service for Apache Flink 可以建立一個持續執行的應用程式、從來源讀取資料、寫入目的地、讓應用程式維持長時間執行狀態，以及根據來源串流的輸送量自動擴展資源。

> **Note**
>
> 應用程式的程式碼匯出到的 S3 儲存貯體必須與 Studio 筆記本位於相同的區域。

只有在符合下列條件的情況下，才能部署 Studio 筆記本的筆記：

- 段落必須按順序排列。當您部署應用程式時，註解中的所有段落都會依序執行 (left-to-right, top-to-bottom)，如同它們出現在您的記事中一樣。您可以透過在筆記中選擇執行所有段落來檢查此順序。
- 你的程式碼是 Python 和 SQL 或 Scala 和 SQL 的組合。我們目前不支持 Python 和斯卡拉在一起的 deploy-as-application.
- 您的筆記必須只包含下列解譯器：`%flink`、`%flink.ssql`、`%flink.pyflink`、`%flink.ipyflink`、`%md`。
- 不支援使用 Zeppelin 內容物件z。不傳回任何結果的方法不會執行任何動作，除記錄警告之外。其他方法將引發 Python 例外狀況或無法在 Scala 中編譯。
- 筆記必須產生單一 Apache Flink 作業。
- 不支援將具有動態資料表的筆記部署為應用程式。
- %md (Markdown) 段落在部署為應用程式時會略過，因為這些段落預期會包含人類可讀的文件，不適合作為產生的應用程式的一部分執行。

- 部署為應用程式時，將會略過不在 Zeppelin 中執行的停用段落。即使停用的段落使用不相容的解譯器 (例如含有`%flink and %flink.ssql` 解譯器的筆記中的 `%flink.ipyflink`)，在將筆記部署為應用程式時，仍會略過該解譯器，並且不會產生錯誤。
- 至少必須有一個段落與原始程式碼 (Flink SQL PyFlink 或 Flink Scala) 一起存在，這些段落已啟用，才能成功執行應用程式部署。
- 在某個段落內的解譯器指令中設定平行處理 (例如 `%flink.ssql(parallelism=32)`) 將在從筆記部署的應用程式中略過。相反地，您可以透過、AWS Command Line Interface或 AWS API 更新已部署的應用程式AWS Management Console，以根據應用程式所需的平行處理原則層級變更平行性和/或 ParallelismPer KPU 設定，或者您也可以為已部署的應用程式啟用自動調度資源。
- 如果要部署為具有持久狀態的應用程式，則您的 VPC 必須具有網際網路存取。如果您的 VPC 無法存取網際網路，請參閱在沒有網際網路存取的 VPC 中部署為具有持久狀態的應用程式。

Scala/Python 條件

- 在 Scala 或 Python 程式碼中，使用 Blink 規劃器 (對於 Scala，是 `senv`，`stenv`；對於 Python，是 `s_env`，`st_env`)，而不是較舊的「Flink」規劃器 (對於 Scala，是 `stenv_2`；對於 Python，是 `st_env_2`)。Apache Flink 專案建議在生產用例中使用 Blink 規劃器，這是Zeppelin 和 Flink 中的預設規劃器。
- Python 段落不得在預定部署為應用程式的筆記中使用使用！的 shell 調用/指派或 IPython 魔術命令，例如 `%timeit` 或 `%conda`。
- 您不能使用 Scala 案例類別作為傳遞給高階資料流程運算子 (如 `map` 和 `filter`) 的函數的參數。如需 Scala 案例類別的相關資訊，請參閱 Scala 文件中的案例類別。

SQL 條件

- 不允許使用簡單的 SELECT 陳述式，因為沒有相當於段落的輸出部分可以傳遞資料。
- 在任何指定段落中，DDL 陳述式 (`USE`、`CREATE`、`ALTER`、`DROP`、`SET`、`RESET`) 都必須放在 DML (`INSERT`) 陳述式前面。這是因為，段落中的 DML 陳述式必須作為單一 Flink 作業一起提交。
- 最多只能有一個段落中包含 DML 陳述式。這是因為，對於此 deploy-as-application 功能，我們只支持向 Flink 提交單個作業。

如需詳細資訊和範例，請參閱搭配使用 SQL 函數與 Amazon Managed Service for Apache Flink、Amazon Translate 和 Amazon Comprehend 來翻譯、修訂和分析串流資料。

Studio 筆記本的 IAM 許可

透過 AWS Management Console 建立 Studio 筆記本時，Managed Service for Apache Flink 會為您建立 IAM 角色。它還會將允許下列存取的政策與該角色相關聯：

服務	存取
CloudWatch 日誌	清單
Amazon EC2	清單
AWS Glue	讀取，寫入
Managed Service for Apache Flink	讀取
Managed Service for Apache Flink V2	讀取
Amazon S3	讀取，寫入

連接器和相依性

連接器可讓您跨越各種技術讀取和寫入資料。Managed Service for Apache Flink 會將三個預設連接器與您的 Studio 筆記本綁定。您也可以使用自訂連接器。如需關於連接器的詳細資訊，請參閱《Apache Flink 文件》中的資料表和 SQL 連接器。

預設連接器

如果您使用 AWS Management Console 建立 Studio 筆記本，Managed Service for Apache Flink 預設會包含下列自訂連接器：`flink-sql-connector-flink`、`flink-connector-kafka_2.12` 和 `aws-msk-iam-auth`。若要在沒有這些自訂連接器的情況下透過主控台建立 Studio 筆記本，請選擇使用自訂設定建立選項。然後，當您進入組態頁面時，清除兩個連接器旁邊的核取方塊。

如果您使用 CreateApplicationAPI 建立 Studio 筆記本，則預設不會包含`flink-sql-connector-flink`和`flink-connector-kafka`連接器。若要新增它們，請將它們指定為 `CustomArtifactsConfiguration` 資料類型 的 `MavenRefernce`，如下列範例所示。

`aws-msk-iam-auth` 連接器是與 Amazon MSK 搭配使用的連接器，其中包含可透過 IAM 自動驗證的功能。

> **Note**
>
> 下列範例中顯示的連接器版本是我們支援的唯一版本。

```
For the Kinesis connector:

"CustomArtifactsConfiguration": [{
"ArtifactType": "DEPENDENCY_JAR",
   "MavenReference": {
"GroupId": "org.apache.flink",

      "ArtifactId": "flink-sql-connector-kinesis",
      "Version": "1.15.4"

   }
}]

For authenticating with AWS MSK through AWS IAM:

"CustomArtifactsConfiguration": [{
"ArtifactType": "DEPENDENCY_JAR",
   "MavenReference": {
"GroupId": "software.amazon.msk",
      "ArtifactId": "aws-msk-iam-auth",
      "Version": "1.1.6"
   }
}]

For the Apache Kafka connector:

"CustomArtifactsConfiguration": [{
"ArtifactType": "DEPENDENCY_JAR",
   "MavenReference": {
"GroupId": "org.apache.flink",

      "ArtifactId": "flink-connector-kafka",
      "Version": "1.15.4"

   }
}]
```

若要將這些連接器新增至現有的筆記本，請使用 UpdateApplicationAPI 作業，並將它們指定為CustomArtifactsConfigurationUpdate資料類型MavenReference中的。

> **Note**
>
> 針對資料表 API 中的 `flink-sql-connector-kinesis` 連接器，您可以將 `failOnError` 設定為 true。

相依性和自訂連接器

若要使用 AWS Management Console 將相依性或自訂連接器新增到 Studio 筆記本，請依照下列步驟：

1. 將自訂連接器的檔案上傳到 Amazon S3。
2. 在 AWS Management Console 中，選擇用於建立 Studio 筆記本的自訂建立選項。
3. 遵循 Studio 筆記本建立工作流程，直到進入組態步驟。
4. 在自訂連接器區段，選擇新增自訂連接器。
5. 指定相依性或自訂連接器的 Amazon S3 位置。
6. 選擇儲存變更。

若要在使用 CreateApplicationAPI 建立新的 Studio 筆記本時新增相依性 JAR 或自訂連接器，請在CustomArtifactsConfiguration資料類型中指定相依性 JAR 的 Amazon S3 位置或自訂連接器。若要將相依性或自訂連接器新增至現有 Studio 筆記本，請呼叫 UpdateApplicationAPI 作業，並在CustomArtifactsConfigurationUpdate資料類型中指定相依性 JAR 或自訂連接器的 Amazon S3 位置。

> **Note**
>
> 包含相依性或自訂連接器時，還必須包含其中未綁定的所有可轉移相依性。

使用者定義的函數

使用者定義的函數 (UDF) 是一些延伸點，可讓您呼叫常用邏輯或無法在查詢中以其他方式表示的自訂邏輯。您可以使用 Python 或類似 Java 或 Scala 的 JVM 語言，在 Studio 筆記本的段落中實作您的 UDF。您也可以將包含以 JVM 語言實作的 UDF 新增至 Studio 筆記本外部 JAR 檔案。

當實作註冊該子類 `UserDefinedFunction` (或您自己的抽像類) 的抽像類的 JAR 時，請使用 Apache Maven 中提供的範圍、Gradle 中的 `compileOnly` 相依性宣告、SBT 中提供的範圍或 UDF 專案建置組態中的等效指令。這可讓 UDF 來源程式碼根據 Flink API 進行編譯，但 Flink API 類別本身並不包含在建置成品中。請參閱來自 UDF jar 範例的此 pom，該範例符合 Maven 專案上的這種先決條件。

> **Note**
>
> 如需範例設定，請參閱 AWS 機器學習部落格中的搭配使用 SQL 函數與 Amazon Managed Service for Apache Flink、Amazon Translate 和 Amazon Comprehend 來翻譯、修訂和分析串流資料。

若要使用主控台將 UDF JAR 檔案新增至您的 Studio 筆記本，請依照下列步驟執行：

1. 將 UDF JAR 檔案上傳至 Amazon S3。
2. 在 AWS Management Console 中，選擇用於建立 Studio 筆記本的自訂建立選項。
3. 遵循 Studio 筆記本建立工作流程，直到進入組態步驟。
4. 在使用者定義的函數區段中，選擇新增使用者定義的函數
5. 指定 JAR 檔案的 Amazon S3 位置，或是具有 UDF 實作的 ZIP 檔案。
6. 選擇儲存變更。

若要在使用 CreateApplicationAPI 建立新的 Studio 筆記本時新增 UDF JAR，請在`CustomArtifactConfiguration`資料類型中指定 JAR 位置。若要將 UDF JAR 新增至現有的 Studio 筆記本，請叫用 UpdateApplicationAPI 作業，並在`CustomArtifactsConfigurationUpdate`資料類型中指定 JAR 位置。您也可以使用 AWS Management Console 將 UDF JAR 檔案新增至 Studio 筆記本。

使用使用者定義函數的考量

- Managed Service for Apache Flink Studio 使用 Apache Zeppelin 術語，其中筆記本是指一個 Zeppelin 執行個體，可以包含多條筆記。然後，每條筆記可以包含多個段落。借助 Managed Service for Apache Flink Studio，解譯器過程在筆記本中的所有筆記間共用。因此，如果您在一個音符中使用 Function 執行明確的createTemporarySystem函數註冊，則可以在同一筆記本的另一個筆記中按原樣引用相同的函數註冊。

然而，部署為應用程式作業只適用於個別筆記，而不是筆記本中的所有筆記。當您執行部署為應用程式時，只會使用作用中筆記的內容來產生應用程式。在其他筆記本中執行的任何明確函數註冊都不屬於產生的應用程式相依性。此外，在使用部署為應用程式選項期間，會透過將 JAR 的主類別名稱轉換為小寫字串，進行隱含函數註冊。

例如，如果 TextAnalyticsUDF 是 UDF JAR 的主類別，則隱含註冊將產生函數名稱 textanalyticsudf。因此，如果 Studio 的筆記 1 中的明確函數註冊如下所示發生，那麼因為共用解譯器，該筆記本中的所有其他筆記 (例如筆記 2) 均可透過名稱 myNewFuncNameForClass 引用該函數：

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

但是，在對筆記 2 執行部署為應用程式操作期間，此明確註冊將不包含在相依性中，因此已部署的應用程式將無法按預期執行。由於隱含註冊，依預設，對此函數的所有引用都應帶有 textanalyticsudf 而不是 myNewFuncNameForClass。

如果需要進行自訂函數名稱註冊，則筆記 2 本身預計將包含另一個段落來執行另一個明確註冊，如下所示：

```
%flink(parallelism=1)
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 UDF JAR 包含 Flink SDK，請設定您的 Java 專案，以便 UDF 來源程式碼可以針對 Flink SDK 進行編譯，但 Flink SDK 類別本身不包含在建置成品中，例如 JAR。

  您可以使用 Apache Maven 中的 provided 範圍、Gradle 中的 compileOnly 相依性宣告、SBT 中的 provided 範圍或 UDF 專案建置組態中的等效指令。您可以參閱 UDF jar 範例中的此 pom，

該範例符合 maven 專案上的這種先決條件。如需完整的 step-by-step 教學課程，請參閱這篇文章，使用 SQL 函數與 Amazon Flink、亞馬遜 Translate 和亞馬遜 CompleAmazon Comprehend 的亞馬遜受管服務來翻譯、編輯和分析串流資料。

啟用檢查點

您可以使用環境設定來啟用檢查點。如需檢查點的相關資訊，請參閱《Managed Service for Apache Flink 開發人員指南》https://docs.aws.amazon.com/managed-flink/latest/java/中的容錯。

設定檢查點間隔

以下 Scala 程式碼範例將應用程式的檢查點間隔設定為 1 分鐘：

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

以下 Python 程式碼範例將應用程式的檢查點間隔設定為 1 分鐘：

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

設定檢查點類型

以下 Scala 程式碼範例將應用程式的檢查點模式設定為 EXACTLY_ONCE (預設值)：

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

以下 Python 程式碼範例將應用程式的檢查點模式設定為 EXACTLY_ONCE (預設值)：

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

使用 AWS Glue

您的 Studio 筆記本會從 AWS Glue 中儲存並取得其資料來源和接收器的相關資訊。建立 Studio 筆記本時，請指定包含連線資訊的 AWS Glue 資料庫。存取資料來源和接收器時，可以指定資料庫中包含

的 AWS Glue 資料表。AWS Glue 資料表可讓您存取定義資料來源和目的地位置、結構描述和參數的 AWS Glue 連線。

Studio 筆記本使用資料表屬性來儲存應用程式特定的資料。如需詳細資訊，請參閱 資料表屬性。

如需如何設定與 Studio 筆記本搭配使用的 AWS Glue 連線、資料庫和資料表的範例，請參閱 建立 Studio 筆記本教學課程 教學課程中的建立 AWS Glue 資料庫。

資料表屬性

除了資料欄位之外，AWS Glue 資料表還會使用資料表屬性為 Studio 筆記本提供其他資訊。Managed Service for Apache Flink 使用下列 AWS Glue 資料表屬性：

- 使用 Apache Flink 時間值：這些屬性定義 Managed Service for Apache Flink 如何發出 Apache Flink 內部資料處理時間值。
- 使用 Flink 連接器和格式屬性：這些屬性提供資料串流的相關資訊。

若要將屬性新增至 AWS Glue 資料表，請執行下列動作：

1. 登入 AWS Management Console，並前往 https://console.aws.amazon.com/glue/ 開啟 AWS Glue 主控台。
2. 從資料表清單中，選擇應用程式用於儲存其資料連線資訊的資料表。依序選擇動作和編輯資料表詳細資訊。
3. 在資料表屬性下，為索引鍵輸入 **managed-flink.proctime**，為值輸入 **user_action_time**。

使用 Apache Flink 時間值

Apache Flink 提供描述何時發生串流處理事件的時間值，例如處理時間和事件時間。若要在應用程式輸出中包含這些值，請定義 AWS Glue 資料表上的屬性，以告知 Managed Service for Apache Flink 執行時間將這些值發送到指定的欄位中。

您在資料表屬性中使用的索引鍵和值如下所示：

Timestamp 類型	金鑰	值
處理時間	managed-flink.proctime	The column name that AWS Glue will use to expose the

Timestamp 類型	金鑰	值
		value. This column name does not correspond to an existing table column.
事件時間	managed-flink.rowtime	The column name that AWS Glue will use to expose the value. This column name corresponds to an existing table column.
	managed-flink.watermark.*column_name*.milliseconds	The watermark interval in milliseconds

使用 Flink 連接器和格式屬性

您可以使用 AWS Glue 資料表屬性向應用程式的 Flink 連接器提供資料來源的相關資訊。Managed Service for Apache Flink 用於連接器的一些屬性範例如下：

連接器類型	金鑰	值
Kafka	##	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. `earliest-offset` or `timestamp` .
Kinesis	##	The format used to deserialize and serialize Kinesis data stream records, e.g. `json` or `csv`.

連接器類型	金鑰	值
	`aws.region`	The AWS region where the stream is defined.
S3 (檔案系統)	format	The format used to deserialize and serialize files, e.g. `json` or `csv`.
	##	The Amazon S3 path, e.g. `s3://mybucket/` .

如需 Kinesis 和 Apache Kafka 以外的其他連接器的相關資訊，請參閱連接器的文件。

範例與教學課程

主題

- 教學課程：在 Managed Service in Apache Flink 中建立 Studio 筆記本
- 教學課程：部署為具有持久狀態的應用程式
- 範例

教學課程：在 Managed Service in Apache Flink 中建立 Studio 筆記本

下列教學課程示範如何建立可從 Kinesis 資料串流或 Amazon MSK 叢集讀取資料的 Studio 筆記本。

本教學課程包含下列章節：

- 設定
- 建立 AWS Glue 資料庫
- 後續步驟
- 使用 Kinesis Data Streams 建立 Studio 筆記本
- 使用 Amazon MSK 建立 Studio 筆記本
- 清理應用程式和相依資源

設定

請確保 AWS CLI 正在執行版本 2 或更新版本。若要安裝最新 AWS CLI，請參閱安裝、更新和解除安裝 AWS CLI 版本 2。

建立 AWS Glue 資料庫

您的 Studio 筆記本使用 AWS Glue 資料庫取得有關 Amazon MSK 資料來源的中繼資料。

建立 AWS Glue 資料庫

1. 前往 https://console.aws.amazon.com/glue/ 開啟 AWS Glue 主控台。
2. 選擇 Add database (新增資料庫)。在新增資料庫視窗中，為資料庫名稱輸入 **`default`**。選擇建立。

後續步驟

借助本教學課程，您可以建立使用 Kinesis Data Streams 或 Amazon MSK 的 Studio 筆記本：

- Kinesis Data Streams：使用 Kinesis Data Streams，您可以快速建立使用 Kinesis 資料串流作為來源的應用程式。您只需要將 Kinesis 資料串流建立為相依資源。
- Amazon MSK：使用 Amazon MSK，您可以建立使用 Amazon MSK 叢集做為來源的應用程式。您需要建立一個 Amazon VPC、一個 Amazon EC2 用戶端執行個體和一個 Amazon MSK 叢集作為相依資源。

使用 Kinesis Data Streams 建立 Studio 筆記本

本教學課程說明如何建立使用 Kinesis 資料串流作為來源的 Studio 筆記本。

本教學課程包含下列章節：

- 設定
- 建立 AWS Glue 資料表
- 使用 Kinesis Data Streams 建立 Studio 筆記本
- 將資料傳送至 Kinesis 資料串流
- 測試 Studio 筆記本

設定

建立 Studio 筆記本之前，請先建立 Kinesis 資料串流 (`ExampleInputStream`)。您的應用程式使用此串流作為應用程式來源。

您可以使用 Amazon Kinesis 主控台或以下 AWS CLI 命令來建立該串流。如需主控台指示，請參閱《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。將該串流命名為 **`ExampleInputStream`**，並將開啟的碎片數量設定為 **`1`**。

若要使用 AWS CLI 建立串流 (`ExampleInputStream`)，請使用以下 Amazon Kinesis `create-stream` AWS CLI 命令。

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

建立 AWS Glue 資料表

您的 Studio 筆記本使用 AWS Glue 資料庫取得有關 Kinesis Data Streams 資料來源的中繼資料。

> **Note**
>
> 您可以先手動建立資料庫，也可以讓 Managed Service for Apache Flink 在您建立筆記本時為您建立資料庫。同樣，您可以依照本節所述手動建立資料表，也可以在 Apache Zeppelin 的筆記本中，使用針對 Managed Service for Apache Flink 的建立資料表連接器程式碼，透過 DDL 陳述式建立資料表。然後，您可以簽入 AWS Glue 以確保資料表已正確建立。

建立資料表

1. 登入 AWS Management Console，並前往 https://console.aws.amazon.com/glue/ 開啟 AWS Glue 主控台。
2. 如果您尚無 AWS Glue 資料庫，請從左側導覽列選擇資料庫。選擇新增資料庫。在新增資料庫視窗中，為資料庫名稱輸入 **`default`**。選擇建立。
3. 在左側導覽窗格中，選擇資料表。在資料表頁面中，選擇新增資料表 > 手動新增資料表。
4. 在設定資料表頁面中，為資料表名稱輸入 **`stock`**。請務必選取先前建立的資料庫。選擇下一步。

5. 在新增資料存放區頁面中，選擇 Kafka。對於串流名稱，輸入 **ExampleInputStream**。針對 Kinesis 來源 URL，請選擇輸入 **https://kinesis.us-east-1.amazonaws.com**。如果您複製並貼上 Kinesis 來源 URL，請務必刪除任何前置或尾端空格。選擇下一步。
6. 在分類頁面中，選擇 JSON。選擇下一步。
7. 在定義結構描述頁面中，選擇「新增資料欄」以新增資料欄。新增具有下列屬性的欄：

欄名稱	資料類型
####	**string**
##	**double**

   選擇下一步。
8. 在下一頁上，確認您的設定，然後選擇完成。
9. 從資料表清單中選取您新建立的資料表。
10. 選擇編輯資料表，然後新增索引鍵為 `managed-flink.proctime` 值為 `proctime` 的屬性。
11. 選擇套用。

使用 Kinesis Data Streams 建立 Studio 筆記本

現在，您已建立應用程式使用的資源，接下來可以建立您的 Studio 筆記本。

您可以使用 AWS Management Console 或 AWS CLI 建立應用程式。

- 使用 AWS Management Console 建立 Studio 筆記本
- 使用 AWS CLI 建立 Studio 筆記本

使用 AWS Management Console 建立 Studio 筆記本

1. 前往 https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard 開啟 Managed Service in the Apache Flink 主控台。
2. 在 Managed Service for Apache Flink 應用程式頁面中，選擇 Studio 標籤。選擇建立 Studio 筆記本。

> **Note**
>
> 您也可以藉由選取輸入 Amazon MSK 叢集或 Kinesis 資料串流，然後選擇即時處理資料，從 Amazon MSK 或 Kinesis Data Streams 主控台建立 Studio 筆記本。

3. 在建立 Studio 筆記本頁面上，提供下列資訊：

   - 為筆記本名稱輸入 **MyNotebook**。
   - 為 AWS Glue 資料庫選擇預設值。

   選擇建立 Studio 筆記本。

4. 在MyNotebook頁面中，選擇 [執行]。等待狀態顯示為執行中。筆記本執行時需支付費用。

使用 AWS CLI 建立 Studio 筆記本

若要使用 AWS CLI 建立 Studio 筆記本，請執行下列動作：

1. 驗證您的帳戶 ID。您需要此值來建立應用程式。
2. 建立角色 `arn:aws:iam::`*`AccountID`*`:role/ZeppelinRole`，並將下列許可新增至主控台自動建立的角色。

   ```
   "kinesis:GetShardIterator",

   "kinesis:GetRecords",

   "kinesis:ListShards"
   ```

3. 建立稱為 `create.json` 的檔案，其中具有以下內容。使用您的帳戶 ID 取代預留位置的值。

   ```
   {
       "ApplicationName": "MyNotebook",
       "RuntimeEnvironment": "ZEPPELIN-FLINK-3_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. 若要建立應用程式，請執行下列命令：

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

5. 命令完成後，您應該會看到類似如下的輸出，其中顯示新 Studio 筆記本的詳細資料：以下為輸出範例。

```
{
    "ApplicationDetail": {
        "ApplicationARN": "arn:aws:kinesisanalyticsus-
east-1:012345678901:application/MyNotebook",
        "ApplicationName": "MyNotebook",
        "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
        "ApplicationMode": "INTERACTIVE",
        "ServiceExecutionRole": "arn:aws:iam::012345678901:role/ZeppelinRole",
...
```

6. 若要執行應用程式，請執行下列命令：以您的帳戶 ID 取代範例值。

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

將資料傳送至 Kinesis 資料串流

若要將測試資料傳送至 Kinesis 資料串流，請執行下列動作：

1. 開啟 Kinesis 資料產生器。
2. 選擇使用建立 Cognito 使用 CloudFormation者。
3. AWS CloudFormation 主控台隨即開啟，其中包含 Kinesis 資料產生器範本。選擇下一步。
4. 在指定元件詳細資訊頁面上，輸入 Cognito 使用者的使用者名稱和密碼。選擇下一步。

5. 在設定堆疊選項頁面，選擇下一步。
6. 在 [檢閱動態資料產生器-認知-使用者] 頁面中，選擇 [我確認可能會建立 IAM 資源]。AWS CloudFormation 核取方塊。選擇 Create Stack (建立堆疊)。
7. 等待 AWS CloudFormation 堆疊建立完成。堆疊完成後，在 AWS CloudFormation 主控台中開啟 Kinesis-Data-Generator-Cognito-User 堆疊，然後選擇 輸出標籤。打開列KinesisDataGeneratorUrl出的輸出值的 URL。
8. 在 Amazon Kinesis 資料產生器頁面中，使用您在步驟 4 中建立的憑證登入。
9. 在下一頁面中，提供下列值：

區域	**us-east-1**
串流/Kinesis Data Firehose 串流	**ExampleInputStream**
每秒記錄數	**1**

為記錄範本貼上下列程式碼：

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

10. 選擇傳送資料。
11. 產生器會將資料傳送至 Kinesis 資料串流。

    完成下一節時，讓產生器保持執行狀態。

測試 Studio 筆記本

在本節中，您可以使用 Studio 筆記本查詢 Kinesis 資料串流中的資料。

1. 前往 https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard 開啟 Managed Service in the Apache Flink 主控台。
2. 在 Managed Service for Apache Flink 應用程式頁面中，選擇 Studio 筆記本標籤。選擇MyNotebook。
3. 在MyNotebook頁面中，選擇在 Apache 齊柏林飛艇中打開。

   Apache Zeppelin 介面會在新標籤中開啟。
4. 在歡迎來到 Zeppelin! 頁面上，選擇 Zeppelin 筆記。
5. 在 Zeppelin 筆記頁面中，在新筆記中輸入以下查詢：

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

   選擇執行圖示。

   一小段時間後，筆記會顯示 Kinesis 資料串流中的資料。

若要為應用程式開啟 Apache Flink 儀表板以檢視操作層面，請選擇 FLINK 作業。如需關於 Flink 儀表板的詳細資訊，請參閱《Managed Service for Apache Flink 開發人員指南》https://docs.aws.amazon.com/中的 Apache Flink 儀表板。

如需 Flink 串流 SQL 查詢的更多範例，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的查詢。

使用 Amazon MSK 建立 Studio 筆記本

本教學課程說明如何建立使用 Amazon MSK 叢集作為來源的 Studio 筆記本。

本教學課程包含下列章節：

- 設定
- 將 NAT 閘道新增至 VPC
- 建立 AWS Glue 連線與表格
- 使用 Amazon MSK 建立 Studio 筆記本
- 將資料傳送至 Amazon MSK 叢集
- 測試 Studio 筆記本

設定

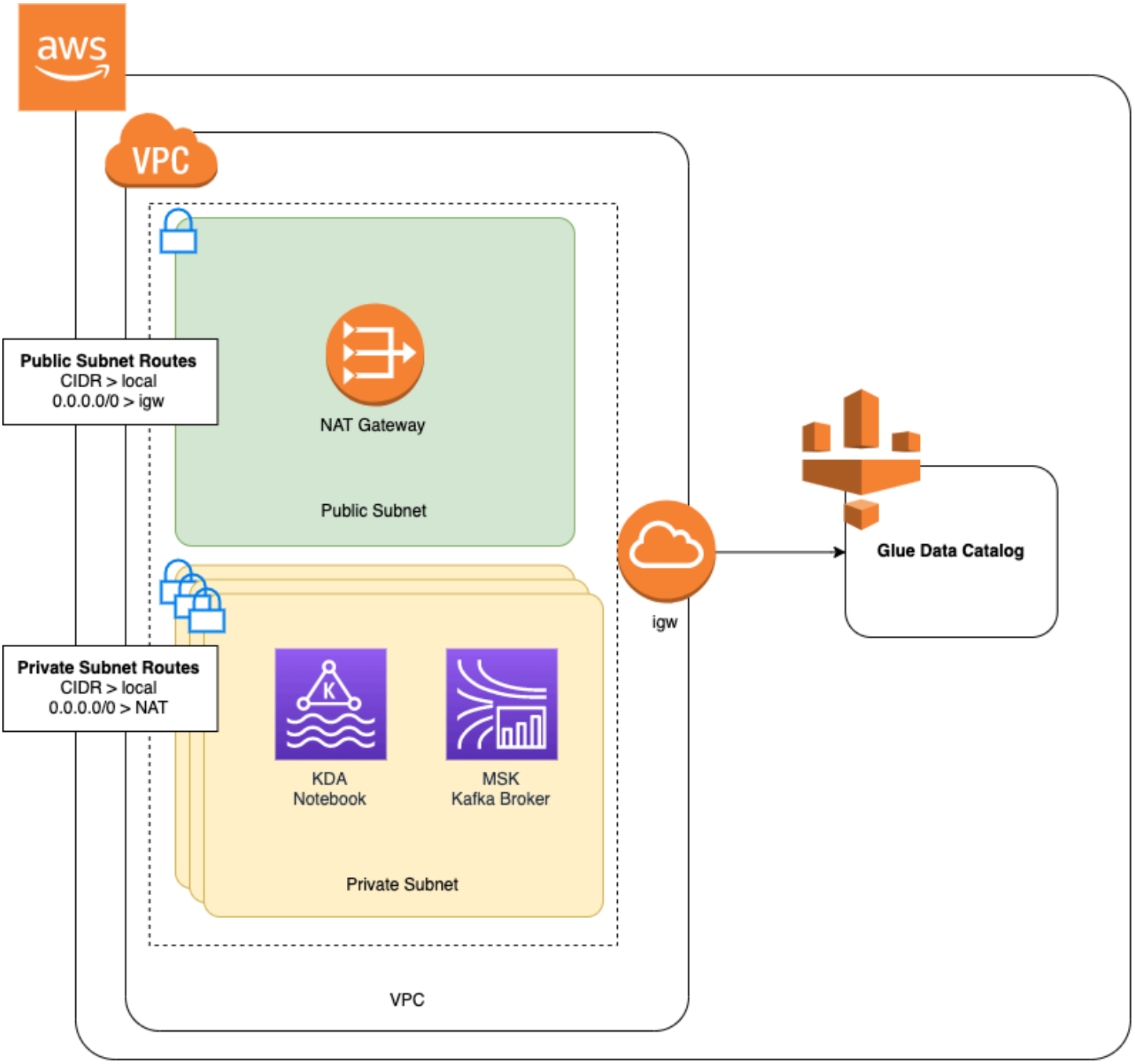
在本教學課程中，您需要一個允許純文字存取的 Amazon MSK 叢集。如果尚未設定 Amazon MSK 叢集，請按照《Amazon MSK 使用入門》https://docs.aws.amazon.com/msk/latest/developerguide/getting-started.html教學課程來建立 Amazon VPC、Amazon MSK 叢集、主題和 Amazon EC2 用戶端執行個體。

跟隨教學課程學習時，請執行下列動作：

- 在步驟 3：建立 Amazon MSK 叢集的步驟 4 中，將 `ClientBroker` 值從 `TLS` 變更為 **`PLAINTEXT`**。

將 NAT 閘道新增至 VPC

如果依照《Amazon MSK 使用入門》https://docs.aws.amazon.com/msk/latest/developerguide/getting-started.html教學課程建立 Amazon MSK 叢集，或者您現有的 Amazon VPC 還沒有適用於其私有子網路的 NAT 閘道，則必須將 NAT 閘道新增到 Amazon VPC。下圖顯示一般架構。



若要為您的 Amazon VPC 建立 NAT 閘道，請執行下列動作：

1. 前往 https://console.aws.amazon.com/vpc/ 開啟 Amazon VPC 主控台。
2. 從左側導覽列選擇 NAT 閘道。
3. 在 NAT 閘道頁面上，選擇建立 NAT 閘道。
4. 在建立 NAT 閘道頁面上，提供下列值：

名稱：選用。	**Zeppelin ##**
子網	AWSKafkaTutorialSubnet1
彈性 ID 配置 ID	Choose an available Elastic IP. If there are no Elastic IPs available, choose 配置彈性 IP, and then choose the Elasic IP that the console creates.

選擇建立 NAT 閘道。

5. 在導覽窗格中，選擇路由表。
6. 選擇建立路由表。
7. 在建立路由表頁面上，提供以下資訊：
   - 名稱標籤：**ZeppelinRouteTable**
   - VPC：選擇 VPC (例如 AWSKafkaTutorialVPC)。

   選擇建立。
8. 在路由表清單中，選擇 ZeppelinRouteTable。選擇路由標籤，然後選擇編輯路由。
9. 在編輯路由標籤中，選擇新增路由。
10. 在 中，為目標輸入 **0.0.0.0/0**。為目標選擇 NAT 閘道、ZeppelinGateway。選擇儲存路由。選擇關閉。
11. 在「路由表」頁面上，已選取 ZeppelinRouteTable 時，選擇子網路關聯標籤。選擇編輯子網路關聯。
12. 在編輯子網路關聯頁面中，選擇 AWSKafkaTutorialSubnet2 和 AWSKafkaTutorialSubnet3。選擇儲存。

建立 AWS Glue 連線與表格

您的 Studio 筆記本使用 AWS Glue 資料庫取得有關 Amazon MSK 資料來源的中繼資料。在本節中，您會建立說明如何存取 Amazon MSK 叢集的 AWS Glue 連線，以及一個說明如何將資料來源中的資料呈現給用戶端 (例如 Studio 筆記本) 的 AWS Glue 表格。

建立連線

1. 登入 AWS Management Console，並前往 https://console.aws.amazon.com/glue/ 開啟 AWS Glue 主控台。
2. 如果您尚無 AWS Glue 資料庫，請從左側導覽列選擇資料庫。選擇新增資料庫。在新增資料庫視窗中，為資料庫名稱輸入 **`default`**。選擇建立。
3. 從左側導覽列選擇程式碼。選擇新增連線。
4. 在新增連線視窗中，提供下列值：

   - 對於連線名稱，請輸入 **`ZeppelinConnection`**。
   - 對於連線類型，請選擇 Kafka。
   - 對於 Kafka 啟動伺服器 URL，請為叢集提供啟動代理程式字串。您可以從 MSK 主控台或輸入下列 CLI 命令來取得啟動代理程式：

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

   - 取消勾選需要 SSL 連線核取方塊。

   選擇下一步。
5. 在 VPC 頁面中，提供下列值：

   - 對於 VPC，請選擇您的 VPC 名稱 (例如 AWSKafkaTutorialVPC)。
   - 對於子網路，請選擇 AWSKafkaTutorialSubnet2。
   - 對於安全性群組，請選擇所有可用的群組。

   選擇下一步。
6. 在連線屬性 / 連線存取權頁面中，選擇完成。

建立資料表

> **Note**
>
> 您可以依照下列步驟所述手動建立資料表，也可以在 Apache Zeppelin 的筆記本中，使用針對 Managed Service for Apache Flink 的建立資料表連接器程式碼，透過 DDL 陳述式建立資料表。然後，您可以簽入 AWS Glue 以確保資料表已正確建立。

1. 在左側導覽窗格中，選擇資料表。在資料表頁面中，選擇新增資料表 > 手動新增資料表。
2. 在設定資料表頁面中，為資料表名稱輸入 **stock**。請務必選取先前建立的資料庫。選擇下一步。
3. 在新增資料存放區頁面中，選擇 Kafka。對於主題名稱，請輸入您的主題名稱（例如 AWSKafkaTutorialTopic）。針對連線，選擇 ZeppelinConnection。
4. 在分類頁面中，選擇 JSON。選擇下一步。
5. 在定義結構描述頁面中，選擇「新增資料欄」以新增資料欄。新增具有下列屬性的欄：

資料欄名稱	資料類型
####	**##**
##	**double**

   選擇下一步。
6. 在下一頁上，確認您的設定，然後選擇完成。
7. 從資料表清單中選取您新建立的資料表。
8. 選擇編輯資料表，然後新增索引鍵為 `managed-flink.proctime` 值為 `proctime` 的屬性。
9. 選擇套用。

使用 Amazon MSK 建立 Studio 筆記本

現在，您已建立應用程式使用的資源，接下來可以建立您的 Studio 筆記本。

您可以使用 AWS Management Console 或 AWS CLI 建立應用程式。

- 使用 AWS Management Console 建立 Studio 筆記本
- 使用 AWS CLI 建立 Studio 筆記本

> **Note**
>
> 您也可以選擇現有叢集，然後選擇即時處理資料，從 Amazon MSK 主控台建立 Studio 筆記本。

使用 AWS Management Console 建立 Studio 筆記本

1. 前往 https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard 開啟 Managed Service in the Apache Flink 主控台。
2. 在 Managed Service for Apache Flink 應用程式頁面中，選擇 Studio 標籤。選擇建立 Studio 筆記本。

   > **Note**
   >
   > 若要從 Amazon MSK 或 Kinesis Data Streams 主控台建立 Studio 筆記本，請選取您的輸入 Amazon MSK 叢集或 Kinesis 資料串流，然後選擇即時處理資料。

3. 在建立筆記本執行個體頁面上，提供下列資訊：

   - 為 Studio 筆記本名稱輸入 **MyNotebook**。
   - 為 AWS Glue 資料庫選擇預設值。

   選擇建立 Studio 筆記本。
4. 在 MyNotebook 頁面中，選擇組態標籤。在網路模式區段中，選擇 VPC。
5. 在編輯 MyNotebook 聯網頁面中，選擇以 Amazon MSK 叢集為基礎的虛擬私人雲端組態。為 Amazon MSK 叢集選擇 Amazon MSK 叢集。選擇儲存變更。
6. 在 MyNotebook 頁面中，選擇執行。等待狀態顯示為執行中。

使用 AWS CLI 建立 Studio 筆記本

若要使用 AWS CLI 建立 Studio 筆記本，請執行下列動作：

1. 請務必備妥下列資訊：您需要這些值來建立應用程式。

   - 帳戶 ID。
   - 子網路 ID 以及包含 Amazon MSK 叢集的 Amazon VPC 的安全群組 ID。
2. 建立稱為 `create.json` 的檔案，其中具有以下內容。使用您的帳戶 ID 取代預留位置的值。

```
{
    "ApplicationName": "MyNotebook",
    "RuntimeEnvironment": "ZEPPELIN-FLINK-3_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. 若要建立應用程式，請執行下列命令：

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

4. 命令完成後，您應該會看到類似如下的輸出，其中顯示新 Studio 筆記本的詳細資料：

```
{
    "ApplicationDetail": {
        "ApplicationARN": "arn:aws:kinesisanalyticsus-
east-1:012345678901:application/MyNotebook",
        "ApplicationName": "MyNotebook",
        "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
        "ApplicationMode": "INTERACTIVE",
        "ServiceExecutionRole": "arn:aws:iam::012345678901:role/ZeppelinRole",
...
```

5. 若要啟動應用程式，請執行下列命令：使用您的帳戶 ID 取代範例值。

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

將資料傳送至 Amazon MSK 叢集

在本節中，您會在 Amazon EC2 用戶端中執行 Python 指令碼，以將資料傳送到您的 Amazon MSK 資料來源。

1. 連線到 Amazon EC2 用戶端。
2. 執行以下命令來安裝 Python 版本 3、Pip 和 Kafka for Python 套件，並確認操作：

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

3. 輸入下列命令，以在您的用戶端電腦上設定 AWS CLI：

```
aws configure
```

   提供帳戶憑證，並為 `region` 提供 **`us-east-1`**。

4. 建立稱為 `stock.py` 的檔案，其中具有以下內容。以 Amazon MSK 叢集的啟動代理程式字串取代範例值，如果您的主題不是 AWSKafkaTutorialTopic，請更新主題名稱：

```
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. 使用下列命令執行指令碼：

```
$ python3 stock.py
```

6. 完成下一節時，讓指令碼保持執行狀態。

測試 Studio 筆記本

在本節中，您可以使用 Studio 筆記本查詢 Amazon MSK 叢集中的資料。

1. 前往 https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard 開啟 Managed Service in the Apache Flink 主控台。
2. 在 Managed Service for Apache Flink 應用程式頁面中，選擇 Studio 筆記本標籤。選擇 MyNotebook。
3. 在 MyNotebook 頁面中，選擇在 Apache Zeppelin 中開啟。

   Apache Zeppelin 介面會在新標籤中開啟。

4. 在歡迎來到 Zeppelin! 頁面上，選擇 Zeppelin 新筆記。
5. 在 Zeppelin 筆記頁面中，在新筆記中輸入以下查詢：

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

選擇執行圖示。

應用程式會顯示 Amazon MSK 叢集中的資料。

若要為應用程式開啟 Apache Flink 儀表板以檢視操作層面，請選擇 FLINK 作業。如需關於 Flink 儀表板的詳細資訊，請參閱《Managed Service for Apache Flink 開發人員指南》https://docs.aws.amazon.com/中的 Apache Flink 儀表板。

如需 Flink 串流 SQL 查詢的更多範例，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的查詢。

清理應用程式和相依資源

刪除 Studio 筆記本

1. 開啟 Managed Service in Apache Flink 主控台。
2. 選擇 MyNotebook。
3. 依序選擇動作和刪除。

刪除 AWS Glue 資料庫和連線

1. 前往 https://console.aws.amazon.com/glue/ 開啟 AWS Glue 主控台。
2. 從左側導覽列選擇資料庫。勾選預設旁邊的核取方塊以選取它。依序選擇動作和刪除資料庫。確認您的選擇。
3. 從左側導覽列選擇連線。選中 ZeppelinConnection 旁邊的核取方塊以選取它。依序選擇動作和刪除連線。確認您的選擇。

刪除 IAM 角色和政策

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 從左側導覽選單選擇角色。

3. 使用搜尋列搜尋 ZeppelinRole 角色。
4. 選擇 ZeppelinRole 角色。選擇刪除角色。確認刪除。

刪除 CloudWatch 日誌群組

使用主控台建立應用程式時，主控台會為您建立 CloudWatch 日誌群組和日誌串流。如果您已使用 AWS CLI 建立應用程式，則沒有日誌群組和串流。

1. 在以下網址開啟 CloudWatch 主控台：https://console.aws.amazon.com/cloudwatch/。
2. 從左側導覽選單選擇日誌群組。
3. 選擇 /AWS/KinesisAnalytics/MyNotebook 日誌群組。
4. 選擇 動作 、刪除日誌群組 。確認刪除。

清除 Kinesis Data Streams 資源

若要刪除 Kinesis 串流，請開啟 Kinesis Data Streams 主控台，選取您的 Kinesis 串流，然後選擇動作 > 刪除。

清除 MSK 資源

如果您已在本教學課程中建立 Amazon MSK 叢集，請按照本節中的步驟進行操作。本節說明如何清理 Amazon EC2 用戶端執行個體、Amazon VPC 和 Amazon MSK 叢集。

刪除 Amazon MSK 叢集

如果您已在本教學課程中建立 Amazon MSK 叢集，請按照以下步驟進行操作。

1. 開啟 Amazon MSK 主控台，網址為 https://console.aws.amazon.com/msk/home?region=us-east-1#/home/。
2. 選擇 AWSKafkaTutorialCluster。選擇 Delete (刪除)。在出現的視窗中輸入 **`delete`**，然後確認選擇。

終止您的用戶端執行個體

如果您已在本教學課程中建立 Amazon EC2 用戶端執行個體，請按照以下步驟進行操作。

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。

2. 從左側導覽窗格中選擇執行個體。
3. 選擇 ZeppelinClient 旁邊的核取方塊以選取它。
4. 依序選擇執行個體狀態和終止執行個體。

刪除 Amazon VPC

如果您已在本教學課程中建立 Amazon VPC，請按照以下步驟進行操作。

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 從左側導覽列選擇網路介面。
3. 在搜尋列中輸入 VPC ID，然後按 Enter 鍵。
4. 選取資料表標題中的核取方塊，以選取所有顯示的網路介面。
5. 選擇 Actions (動作)、Detach (分離)。在出現的視窗中，選擇強制分離下方的啟用。選擇分離，然後等待所有網路介面都到達可用狀態。
6. 選取資料表標題中的核取方塊，以再次選取所有顯示的網路介面。
7. 選擇 動作 、刪除 。確認動作。
8. 前往 https://console.aws.amazon.com/vpc/ 開啟 Amazon VPC 主控台。
9. 選取支持的 AWSVKafkaTutorialVPC。依序選擇動作和刪除。輸入 **`delete`** 並確認刪除。

教學課程：部署為具有持久狀態的應用程式

下列教學課程示範如何將 Studio 筆記本部署為具有持久狀態的 Managed Service for Apache Flink 應用程式。

本教學課程包含下列章節：

- 設定
- 使用 AWS Management Console 部署具有持久狀態的應用程式
- 使用 AWS CLI 部署具有持久狀態的應用程式

設定

按照建立 Studio 筆記本教學課程建立新的 Studio 筆記本，使用 Kinesis Data Streams 或 Amazon MSK。命名 Studio 筆記本 `ExampleTestDeploy`。

使用 AWS Management Console 部署具有持久狀態的應用程式

1. 在主控台中的應用程式程式碼位置 - 選用下，新增您希望將封裝程式碼存放到的 S3 儲存貯體位置。這可讓步驟直接從筆記本部署和執行應用程式。
2. 將必要的許可新增至應用程式角色，以啟用您要用來讀取和寫入 Amazon S3 儲存貯體以及啟動 Managed Service for Apache Flink 應用程式的角色：
   - AmazonS3FullAccess
   - Amazonmanaged-flinkFullAccess
   - 視情況存取您的來源、目的地和 VPC。如需詳細資訊，請參閱Studio 筆記本的 IAM 許可。
3. 請參閱以下範例程式碼：

```
%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. 啟動此功能後，您將在筆記本中每條筆記的右上角看到一個新的下拉式選單，其中包含筆記本的名稱。您可以執行下列作業：
   - 在 AWS Management Console 中檢視 Studio 筆記本設定。
   - 建立 Zeppelin 筆記，並將其匯出至 Amazon S3。此時，請為您的應用程式提供名稱，然後選擇建置和匯出。匯出完成後，您會收到通知。
   - 如有需要，您可以在 Amazon S3 中的可執行檔上檢視和執行任何其他測試。
   - 建置完成後，您將能夠將程式碼部署為具有持久狀態和自動調度資源的 Kinesis 串流應用程式。
   - 使用下拉式選單並選擇將 Zeppelin 筆記部署為 Kinesis 串流應用程式。檢閱應用程式名稱，然後選擇透過 AWS 主控台部署。

- 這將引導您前往建立 Managed Service for Apache Flink 應用程式的 AWS Management Console 頁面。請注意，已預先填入應用程式名稱、平行處理層級、程式碼位置、預設 Glue DB、VPC (如果適用) 和 IAM 角色。驗證 IAM 角色是否具有來源和目的地的必要許可。快照預設啟用，以進行持久的應用程式狀態管理。
- 選擇建立應用程式。
- 您可以選擇設定並修改任何設定，然後選擇執行以啟動串流應用程式。

使用 AWS CLI 部署具有持久狀態的應用程式

若要使用 AWS CLI 部署應用程式，您必須更新 AWS CLI 以使用 Beta 2 資訊隨附的服務模型。如需如何使用更新的服務模型之詳細資訊，請參閱 設定。

以下範例程式碼會建立 Studio 筆記本：

```
aws kinesisanalyticsv2 create-application \
     --application-name <app-name> \
     --runtime-environment ZEPPELIN-FLINK-3_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
```

下列程式碼範例會啟動 Studio 筆記本：

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

下列程式碼會傳回應用程式的 Apache Zeppelin 筆記本頁面的 URL：

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

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

範例

下列範例查詢示範如何在 Studio 筆記本中使用視窗查詢來分析資料。

- 使用 Amazon MSK/Apache Kafka 建立資料表
- 使用 Kinesis 建立資料表
- 輪轉視窗
- 滑動視窗
- 互動式 SQL
- BlackHole SQL 連接器

- 資料產生器
- 互動式 Scala
- 互動式 Python
- 互動式 Python、SQL 和 Scala
- 跨帳戶 Kinesis 資料串流

如需關於 Apache Flink SQL 查詢設定的資訊，請參閱在 Zeppelin 筆記本上使用 Flink 進行互動式資料分析。

若要在 Apache Flink 儀表板中檢視應用程式，請在應用程式的Zeppelin 筆記頁面中選擇 FLINK 作業。

如需視窗查詢的詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的視窗。

如需 Apache Flink 串流 SQL 查詢的更多範例，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的查詢。

使用 Amazon MSK/Apache Kafka 建立資料表

您可以將 Amazon MSK Flink 連接器與 Managed Service for Apache Flink Studio 搭配使用，以使用純文字、SSL 或 IAM 身分驗證來驗證您的連線。根據您的需求，使用特定屬性建立資料表。

```
-- Plaintext connection

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

-- SSL connection

CREATE TABLE your_table (
  `column1` STRING,
  `column2` BIGINT
```

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

-- IAM connection (or for MSK Serverless)

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

您可以將這些與 Apache Kafka SQL 連接器的其他屬性結合使用。

使用 Kinesis 建立資料表

下列範例示範如何使用 Kinesis 建立資料表：

```
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'
);
```

如需可使用的其他屬性的詳細資訊，請參閱 Amazon Kinesis Data Streams SQL 連接器。

輪轉視窗

下列 Flink 串流 SQL 查詢會從 ZeppelinTopic 資料表中選取每個五秒鐘輪轉時段中的最高價格：

```
%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)
```

滑動視窗

下列 Apache Flink 串流 SQL 查詢會從 ZeppelinTopic 資料表中選取每個五秒滑動視窗中的最高價格：

```
%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 AWS Glue
GROUP BY HOP(event_time, INTERVAL '3' SECOND, INTERVAL '5' SECOND)
```

互動式 SQL

此範例會列印事件時間和處理時間的最大值，以及索引鍵-值資料表中的值的總和。請確定您擁有 the section called “資料產生器” 執行中的範例資料產生指令碼。若要在您的 Studio 筆記本中嘗試其他 SQL 查詢 (例如篩選和聯結)，請參閱《Apache Flink 文件》中的查詢。

```
%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 連接器

BlackHole SQL 連接器不需要您建立 Kinesis 資料串流或 Amazon MSK 叢集來測試查詢。如需 BlackHole SQL 連接器的相關資訊，請參閱 Apache Flink 說明文件中的 BlackHole SQL 連接器。在此範例中，預設目錄是記憶體內目錄。

```
%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
```

資料產生器

此範例使用 Scala 生成範例資料。您可以使用此範例資料測試各種查詢。使用 create table 陳述式來建立索引鍵-值資料表。

```
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`
```

互動式 Scala

這是 the section called “互動式 SQL” 的 Scala 翻譯。如需更多 Scala 範例，請參閱《Apache Flink 文件》中的資料表 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`
```

互動式 Python

這是 the section called “互動式 SQL” 的 Python 翻譯。如需更多 Python 範例，請參閱《Apache Flink 文件》中的資料表 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`
```

互動式 Python、SQL 和 Scala

您可以在筆記本中使用 SQL、Python 和 Scala 的任意組合進行互動式分析。在您計劃部署為具有持久狀態的應用程式的 Studio 筆記本中，可以使用 SQL 和 Scala 的組合。此範例顯示略過的區段，以及在應用程式中部署為持久狀態的區段。

```
%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!")
```

跨帳戶 Kinesis 資料串流

若要使用擁有您 Studio 筆記本之帳戶以外的帳戶中的 Kinesis 資料串流，請在執行 Studio 筆記本的帳戶中建立服務執行角色，在具有資料串流的帳戶中建立角色信任政策。在 Kinesis 連接器中的 create table DDL 陳述式中，使用 `aws.credentials.provider`、`aws.credentials.role.arn` 和 `aws.credentials.role.sessionName`，針對資料串流建立資料表。

對 Studio 筆記本帳戶使用下列服務執行角色。

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

對資料串流帳戶使用`AmazonKinesisFullAccess` 政策和下列角色信任政策。

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

為 create table 陳述式使用下面的段落。

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

故障診斷

本節包含 Studio 筆記本的疑難排解資訊。

停止停滯的應用程式

若要停止停留在暫時狀態的應用程式，請在`Force`參數設定為的情況下呼叫StopApplication動作`true`。如需詳細資訊，請參閱《Managed Service for Apache Flink 開發人員指南》https://docs.aws.amazon.com/managed-flink/latest/java/中的執行應用程式。

在沒有網際網路存取的 VPC 中部署為具有持久狀態的應用程式

Apache Flink 工作室 deploy-as-application 功能的託管服務不支持沒有互聯網訪問的 VPC 應用程序。我們建議您在 Studio 中建置應用程式，然後使用 Managed Service for Apache Flink 手動建立 Flink 應用程式，並選取您在筆記本中建置的 zip 檔案。

下列步驟概述了這種方法：

1. 建置 Studio 應用程式並將其匯出到 Amazon S3。這必須是一個 zip 檔案。
2. 使用引用 Amazon S3 中 zip 檔案位置的程式碼路徑，手動建立 Managed Service for Apache Flink 應用程式。此外，您將需要使用以下 env 變數 (總共 2 個 `groupID`，3 個 `var`) 設定應用程式：

3. kinesis.analytics.flink.run.options

    a. python: source/note.py

    b. 罐子文件：庫PythonApplicationDependencies/.

4. managed.deploy_as_app.options

    - DatabaseARN: *<glue database ARN (Amazon Resource Name)>*

5. 您可能需要為應用程式使用的服務授予使用 Managed Service for Apache Flink Studio 和 Managed Service for Apache Flink IAM 角色的許可。您可以為兩個應用程式使用相同的 IAM 角色。

D eploy-as-app 尺寸和構建時間縮短

Python 應用程序的工作室 deploy-as-app 打包 Python 環境中可用的所有內容，因為我們無法確定您需要哪些庫。這可能會導致比必要的大小。 deploy-as-app 下列程序示範如何藉由解除安裝相依性來減少 deploy-as-app Python 應用程式大小的大小。

如果您正在構建具有 Studio deploy-as-app 功能的 Python 應用程序，如果您的應用程序不依賴，則可以考慮從系統中刪除預先安裝的 Python 包。這不僅有助於減少最終成品大小，以避免違反應用程序大小的服務限制，還可以縮短具有該 deploy-as-app 功能的應用程序的構建時間。

您可以執行以下命令以列出所有已安裝的 Python 套件及其各自的安裝大小，並有選擇地移除較大的套件。

```
%flink.pyflink

!pip list --format freeze | awk -F = {'print $1'} | xargs pip show | grep -E
 'Location:|Name:' | cut -d ' ' -f 2 | paste -d ' ' - - | awk '{gsub("-","_",$1); print
 $2 "/" tolower($1)}' | xargs du -sh 2> /dev/null | sort -hr
```

> **Note**
>
> apache-beam 是 Flink Python 運作必需的套件。始終不能移除此套件及其相依項。

以下是在 Studio V2 中預先安裝的、可以考慮移除的 Python 套件之清單：

```
scipy
```

```
statsmodels
plotnine
seaborn
llvmlite
bokeh
pandas
matplotlib
botocore
boto3
numba
```

若要從 Zeppelin 筆記本中移除 Python 套件：

1. 在移除之前，請檢查您的應用程式是否依賴於該套件或其任何消費套件。您可以使用 pipdeptree 識別套件的相依性。
2. 執行以下命令來移除套件：

```
%flink.pyflink
!pip uninstall -y <package-to-remove>
```

3. 如果需要檢索錯誤移除的套件，請執行以下命令：

```
%flink.pyflink
!pip install <package-to-install>
```

Example 示例：在使用功能部署 Python 應用程序之前刪除**scipy** deploy-as-app 包。

1. 使用 pipdeptree 發現所有 scipy 使用者，並驗證是否可以安全移除 scipy。
   - 透過筆記本安裝該工具：

```
%flink.pyflink
!pip install pipdeptree
```

   - 藉由執行以下命令取得 scipy 的反向相依性樹：

```
%flink.pyflink
!pip -r -p scipy
```

   您應該會看到類似下列的輸出 (為求簡化已進行壓縮)：

```
...
------------------------------------------------------------------------
scipy==1.8.0
### plotnine==0.5.1 [requires: scipy>=1.0.0]
### seaborn==0.9.0 [requires: scipy>=0.14.0]
### statsmodels==0.12.2 [requires: scipy>=1.1]
    ### plotnine==0.5.1 [requires: statsmodels>=0.8.0]
```

2. 仔細檢查 seaborn、statsmodels 和 plotnine 在應用程式中的使用情況。如果應用程式不依賴 scipy、seaborn、statemodels 或 plotnine 中的任意一項，便可移除所有這些套件，或只移除應用程式不需要的套件。
3. 執行以下命令移除套件：

```
!pip uninstall -y scipy plotnine seaborn statemodels
```

取消作業

本節說明如何取消無法從 Apache Zeppelin 取得的 Apache Flink 作業。若要取消此類作業，請前往 Apache Flink 儀表板，複製作業 ID，然後在下列其中一個範例中使用它。

取消單一作業：

```
%flink.pyflink
import requests

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

取消所有執行中作業：

```
%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))
```

取消所有作業：

```
%flink.pyflink
import requests

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

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

重新啟動 Apache Flink 解譯器

在 Studio 筆記本中重新啟動 Apache Flink 解譯器

1. 選擇熒幕右上角附近的組態。
2. 選擇解譯器。
3. 選擇重新啟動，然後按確定。

附錄：建立自訂 IAM 政策

您通常會使用受管 IAM 政策來允許應用程式存取相依資源。如果需要更好地控制應用程式的許可，可以使用自訂 IAM 政策。本節包含自訂 IAM 政策的範例。

> **Note**
>
> 在下列政策範例中，以應用程式的值取代預留位置文字。

本主題包含下列章節：

- AWS Glue
- CloudWatch 日誌
- Kinesis 串流
- Amazon MSK 叢集

AWS Glue

下列範例政策授予存取 AWS Glue 資料庫的許可。

```
{
    "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 日誌

下列原則授與存取 CloudWatch 記錄檔的權限：

```
{
      "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**
>
> 如果您使用主控台建立應用程式，則主控台會新增必要的原則，以存取應用程式角色的 CloudWatch 記錄檔。

Kinesis 串流

應用程式可以將 Kinesis 串流用於來源或目的地。應用程式需要讀取許可才能從來源串流讀取，需要寫入許可才能寫入目的地串流。

下列政策授予從用作來源的 Kinesis 串流讀取的許可：

```
{
```

```
  "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/*"
    }
  ]
}
```

下列政策授予向用作目的地的 Kinesis 串流寫入的許可：

```
{
    "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>"
        }
    ]
}
```

如果應用程式存取加密的 Kinesis 串流，則必須授予額外的許可，以存取該串流及其加密金鑰。

下列政策授予存取加密來源的串流和及其加密金鑰的許可：

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

下列政策授予存取加密目的地的串流和及其加密金鑰的許可：

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

Amazon MSK 叢集

若要授予 Amazon MSK 叢集的存取權，可以授予叢集 VPC 的存取權。如需存取 Amazon VPC 的政策範例，請參閱 VPC 應用程式許可。

開始使用適用於阿帕奇 Flink 的 Amazon 託管服務 (DataStream API)

本節將為您介紹適用於 Apache Flink 的受管理服務和 DataStream API 的基本概念。它描述了建立和測試應用程式的可用選項。此外，它還提供了相關指示，以協助您安裝完成本指南教學課程以及建立您的第一個應用程式所需要的工具。

主題

Managed Service for Apache Flink 應用程式的元件

為了處理資料，您的 Managed Service for Apache Flink 應用程式使用 Java/Apache Maven 或 Scala 應用程式來處理輸入，使用 Apache Flink 執行時間生成輸出。

Managed Service for Apache Flink 應用程式包含下列元件：

- 執行時間屬性：您可以使用執行時間屬性來設定應用程式，無需重新編譯應用程式的程式碼。
- 來源：應用程式使用來源來消耗資料。來源連接器從 Kinesis 資料串流、Amazon S3 儲存貯體等讀取資料。如需詳細資訊，請參閱來源。
- 運算子：應用程式會使用一或多個運算子來處理資料。運算子可以轉換、富集或彙總資料。如需詳細資訊，請參閱DataStream API 運算子。
- 接收器：應用程式透過使用接收器生成資料到外部來源。接收器連接器會將資料寫入 Kinesis 資料串流、Kinesis Data Firehose 串流、Amazon S3 儲存貯體等。如需詳細資訊，請參閱接收。

建立、編譯和封裝應用程式的程式碼後，將程式碼套件上傳到 Amazon Simple Storage Service (Amazon S3) 儲存貯體。然後建立 Managed Service for Apache Flink 應用程式。您可以傳入程式碼套

件位置，Kinesis 資料串流作為串流資料來源，以及通常是接收應用程式處理後的資料的串流或檔案位置。

完成練習的先決條件

若要完成本指南中的步驟，您必須執行下列各項：

- Java 開發套件 (JDK) 版本 11。將 JAVA_HOME 環境變數設為指向您的 JDK 安裝位置。
- 我們建議您使用開發環境 (如 Eclipse Java Neon 或 IntelliJ Idea) 來開發和編譯您的應用程式。
- Git 用戶端。如果您尚未安裝 Git 用戶端，請先完成安裝。
- Apache Maven 編譯器外掛程式。Maven 必須在您的工作路徑中。若要測試您的 Apache Maven 安裝，輸入以下資訊：

```
$ mvn -version
```

開始執行，請移至 步驟 1：設定 AWS 帳戶並建立管理員使用者。

步驟 1：設定 AWS 帳戶並建立管理員使用者

第一次使用 Managed Service for Apache Flink 之前，請先完成以下任務：

註冊 AWS 帳戶

如果您還沒有 AWS 帳戶，請完成以下步驟建立新帳戶。

註冊 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   註冊 AWS 帳戶時，會建立 AWS 帳戶根使用者。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為最佳安全實務，將管理存取權指派給管理使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

註冊程序完成後，AWS 會傳送一封確認電子郵件給您。您可以隨時登錄 https://aws.amazon.com/ 並選擇 我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立管理使用者

當您註冊 AWS 帳戶 之後，請保護您的 AWS 帳戶根使用者，啟用 AWS IAM Identity Center，並建立管理使用者，讓您可以不使用根使用者處理日常作業。

保護您的 AWS 帳戶根使用者

1. 選擇 根使用者 並輸入您的 AWS 帳戶電子郵件地址，以帳戶擁有者身分登入 AWS Management Console。在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入使用者指南中的以根使用者身分登入。
2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶根使用者啟用虛擬 MFA 裝置 (主控台)。

建立管理使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，將管理權限授予管理使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的教學課程，請參閱《使用 AWS IAM Identity Center 使用者指南中的以預設 IAM Identity Center 目錄 設定使用者存取權限。

以管理員的身分登入

- 若要使用您的 IAM 身分中心使用者登入，請使用建立 IAM 身分中心使用者時傳送至您電子郵件地址的登入 URL。

  如需有關如何使用 IAM Identity Center 使用者登入的說明，請參閱《AWS 登入 使用者指南》中的登入 AWS存取入口網站。

授予程式設計存取權

若使用者想要與 AWS Management Console 之外的 AWS 互動，則需要程式設計存取權。授予程式設計存取權的方式取決於存取 AWS 的使用者類型。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 設定 AWS CLI 來使用 AWS IAM Identity Center。 • 關於 AWS SDKs、工具和 AWS APIs，請參閱 AWSSDKs 和工具參考指南 中的 IAM Identity Center 驗證。
IAM	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請遵循 IAM 使用者指南 中 使用臨時憑證搭配 AWS 資源 中的指示。
IAM	(不建議使用) 使用長期憑證簽署 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 使用 IAM 使用者憑證進行驗證。 • 關於 AWS SDKs 和工具，請參閱 AWSSDKs 和工具參考

哪個使用者需要程式設計存取權？	到	By
		指南 中的 使用長期憑證進行驗證。 • 關於 AWS API，請參閱 IAM 使用者指南 中的 管理 IAM 使用者的存取金鑰。

後續步驟

步驟 2：設定 AWS Command Line Interface (AWS CLI)

步驟 2：設定 AWS Command Line Interface (AWS CLI)

在此步驟中，您將下載並設定與 Amazon Managed Service for Apache Flink 搭配使用的 AWS CLI。

> **Note**
>
> 本指南的入門練習均假設您使用帳戶中的管理員登入資料 (adminuser) 來執行操作。

> **Note**
>
> 如果您已經安裝 AWS CLI，則可能需要升級才能取得最新功能。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的安裝 AWS Command Line Interface。若要查看 AWS CLI 的版本，執行以下命令：
>
> ```
> aws --version
> ```
>
> 此教學課程中的練習需要以下 AWS CLI 版本或更新版本：
>
> ```
> aws-cli/1.16.63
> ```

設定 AWS CLI

1. 下載和設定 AWS CLI。如需說明，請參閱《AWS Command Line Interface 使用者指南》中的下列主題：

   - 安裝 AWS Command Line Interface
   - 設定 AWS CLI

2. 在 AWS CLIconfig 檔案中，為管理員使用者新增具名設定檔。當您執行 AWS CLI 命令時，使用此設定檔。如需具名描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的具名描述檔。

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

   如需可用 AWS 區域的清單，請參閱 Amazon Web Services 一般參考 中的 區域與端點。

   > **Note**
   >
   > 本教學課程中的範例程式碼和命令使用美國西部 (奧勒岡) 區域。若要使用其他區域，請將本教學課程的程式碼和指令中的「區域」變更為您要使用的區域。

3. 在命令提示字元中輸入下列 help 命令，以驗證設定：

   ```
   aws help
   ```

設定AWS帳戶之後AWS CLI，您可以嘗試下一個練習，在其中設定範例應用程式並測試 end-to-end 設定。

後續步驟

步驟 3：建立並執行 Managed Service for Apache Flink 應用程式

步驟 3：建立並執行 Managed Service for Apache Flink 應用程式

在本練習中，您會建立 Managed Service for Apache Flink 應用程式，並將資料串流作為來源和目的地。

本節包含下列步驟：

建立兩個 Amazon Kinesis Data Streams

在為本練習建立 Managed Service for Apache Flink 應用程式之前，請先建立兩個 Kinesis 資料串流 (`ExampleInputStream` 和 `ExampleOutputStream`)。您的應用程式會將這些串流用於應用程式來源和目的地串流。

您可以使用 Amazon Kinesis 主控台或以下 AWS CLI 命令來建立這些串流。如需主控台指示，請參閱《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。

建立資料串流 (AWS CLI)

1. 若要建立第一個串流 (`ExampleInputStream`)，請使用以下 Amazon Kinesis `create-stream` AWS CLI 命令。

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

2. 若要建立應用程式用來寫入輸出的第二個串流，請執行相同的命令，將串流名稱變更為 `ExampleOutputStream`。

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

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

1. 使用下列內容建立名為 stock.py 的檔案：

```
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. 在教學課程後半段，您會執行 stock.py 指令碼來傳送資料至應用程式。

```
$ python stock.py
```

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的 Java 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，執行下列操作：

1. 使用以下指令複製遠端儲存庫：

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

2. 請前往 amazon-kinesis-data-analytics-java-examples/GettingStarted 目錄。

請留意下列與應用程式的程式碼相關的資訊：

- 專案物件模型 (pom.xml) 檔案包含應用程式的組態和相依性資訊，包括 Managed Service for Apache Flink 程式庫。
- BasicStreamingJob.java 檔案包含定義應用程式功能的 main 方法。
- 應用程式使用 Kinesis 來源從來源串流讀取。以下程式碼片段會建立 Kinesis 來源：

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

- 您的應用程式會建立來源與目的地連接器，以使用 StreamExecutionEnvironment 物件來存取外部資源。
- 應用程式會使用靜態屬性來建立來源與目的地連接器。若要使用動態應用程式屬性，請使用 createSourceFromApplicationProperties 和 createSinkFromApplicationProperties 方法來建立連接器。這些方法會讀取應用程式的屬性，來設定連接器。

  如需執行時間屬性的詳細資訊，請參閱執行時間屬性。

編譯應用程式的程式碼

在本節中，您會使用 Apache Maven 編譯器來建立應用程式的 Java 程式碼。如需安裝 Apache Maven 和 Java 開發套件 (JDK) 的詳細資訊，請參閱 完成練習的先決條件。

編譯應用程式的程式碼

1. 請將應用程式的程式碼編譯並封裝成 JAR 檔案，以使用應用程式的程式碼。您可以使用下列兩種方式的其中之一，編譯和封裝您的程式碼：

   - 使用命令列 Maven 工具。請在包含 `pom.xml` 檔案的目錄中執行下列命令，來建立 JAR 檔案：

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

   - 設定開發環境。如需詳細資訊，請參閱您的開發環境文件。

     > **Note**
     >
     > 提供的來源程式碼依賴於 Java 11 中的程式庫。

   您可以將您的套件做為 JAR 檔案上傳，或壓縮您的套件並做為 ZIP 檔案上傳。如果是使用 AWS CLI 建立應用程式，您會指定程式碼內容類型 (JAR 或 ZIP)。

2. 如果編譯時發生錯誤，請確認您的 `JAVA_HOME` 環境變數是否正確設定。

如果應用程式成功編譯，則會建立下列檔案：

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

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會建立 Amazon Simple Storage Service (Amazon S3) 儲存貯體並上傳您的應用程式的程式碼。

上傳應用程式的程式碼

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 選擇建立儲存貯體。
3. 在儲存貯體名稱欄位中，輸入 **ka-app-code-*<username>***。新增尾碼至儲存貯體名稱，例如您的使用者名稱，使其成為全域唯一的。選擇下一步。
4. 在設定選項步驟中，保留原有設定並選擇 Next (下一步)。
5. 在設定許可步驟中，保留原有設定並選擇 Next (下一步)。

6. 選擇建立儲存貯體。
7. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
8. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `aws-kinesis-analytics-java-apps-1.0.jar` 檔案。選擇下一步。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

您可以使用主控台或 AWS CLI 建立和執行 Managed Service for Apache Flink 應用程式。

> **Note**
>
> 使用主控台建立應用程式時，系統會為您建立 AWS Identity and Access Management (IAM) 和 Amazon CloudWatch 日誌資源。當您使用 AWS CLI 建立應用程式時，則會個別建立這些資源。

主題

建立和執行應用程式 (主控台)

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **`MyApplication`**。
   - 對於 Description (說明)，輸入 **`My java test app`**。
   - 針對執行時間，選擇 Apache Flink。

- 將版本下拉式清單保留為 Apache Flink 1.15.2 版 (建議版本)。

4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：kinesis-analytics-service-*MyApplication*-*us-west-2*
> - 角色：kinesisanalytics-*MyApplication*-*us-west-2*

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **aws-kinesis-analytics-java-apps-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在 Properties (屬性) 下，針對 Group ID (群組 ID)，輸入 **ProducerConfigProperties**。
5. 輸入以下應用程式屬性和數值：

群組 ID	金鑰	值
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

6. 在監控下，確保監控指標層級設為應用程式。
7. 若要CloudWatch 記錄，請選取 [啟用] 核取方塊。
8. 選擇更新。

> **Note**
>
> 當您選擇啟用 Amazon 日誌 CloudWatch 記錄時，Apache Flink 的受管服務會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

停止應用程式

在MyApplication頁面上，選擇 [停止]。確認動作。

更新應用程式

您可以使用主控台更新應用程式設定，例如應用程式屬性、監控設定及位置或應用程式 JAR 的檔名。如果需要更新應用程式的程式碼，也可以從 Amazon S3 儲存貯體重新載入應用程式 JAR。

在MyApplication頁面上，選擇設定。更新應用程式設定，然後選擇更新。

建立並執行應用程式 (AWS CLI)

在本節中，您可以使用 AWS CLI 建立和執行 Managed Service for Apache Flink 應用程式。Managed Service for Apache Flink 使用 `kinesisanalyticsv2` AWS CLI 命令建立 Managed Service for Apache Flink 應用程式並與之互動。

建立許可政策

> **Note**
>
> 您必須為應用程式建立許可政策和角色。如果您未建立這些 IAM 資源，應用程式將無法存取其資料和日誌串流。

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 `read` 動作的許可，而另一條則是授與目的地串流上 `write` 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因

此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

使用以下程式碼來建立 AKReadSourceStreamWriteSinkStream 許可政策。以您用於建立 Amazon S3 儲存貯體 (以儲存應用程式的程式碼) 的使用者名稱來取代 *username*。以您的帳戶 ID 取代 Amazon Resource Names (ARNs) (*012345678901*) 中的帳戶 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"
        }
    ]
}
```

如需建立許可政策的指 step-by-step 示，請參閱 IAM 使用指南中的教學課程：建立和附加您的第一個客戶受管政策。

> **Note**
>
> 若要存取其他 Amazon 服務，您可以使用 AWS SDK for Java。Managed Service for Apache Flink 自動將 SDK 所需的憑證設定為與應用程式相關聯的服務執行 IAM 角色。無須採取額外的步驟。

建立 IAM 角色

在本節中，您會建立 Managed Service for Apache Flink 應用程式可以擔任的 IAM 角色，以便讀取來源串流與寫入目的地串流。

Managed Service for Apache Flink 沒有許可，無法存取串流。您可以透過 IAM 角色來授與這些許可。各 IAM 角色都有連接兩項政策。信任政策會授與擔任角色的 Managed Service for Apache Flink 許可，而許可政策決定了 Managed Service for Apache Flink 在擔任角色後可以執行的作業。

您會將在上一節中建立的許可政策連接至此角色。

若要建立一個 IAM 角色

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Roles (角色) 、Create Role (建立角色)。
3. 在選取可信身分類型下，選擇 AWS 服務。在 Choose the service that will use this role (選擇將使用此角色的服務) 下，選擇 Kinesis。在 Select your use case (選取您的使用案例) 下，選擇 Kinesis Analytics (Kinesis 分析)。

   選擇 Next: Permissions (下一步：許可)。
4. 在 Attach permissions policies (連接許可政策) 頁面上，選擇 Next: Review (下一步：檢閱)。您會在建立角色後連接許可政策。
5. 在建立角色頁面，輸入 **`MF-stream-rw-role`** 作為角色名稱。選擇建立角色。

   現在您已建立新的 IAM 角色，名為 `MF-stream-rw-role`。您接著會更新角色的信任和許可政策。
6. 將 許可政策連接到角色。

> **Note**
>
> 在此練習中，Managed Service for Apache Flink 擔任從 Kinesis 資料串流 (來源) 讀取資料並將輸出寫入另一個 Kinesis 資料串流的角色。因此您會連接在上一個步驟中建立的政策，the section called “建立許可政策”。

a. 在摘要頁面上，選擇許可標籤。
b. 選擇 Attach Policies (連接政策)。
c. 在搜尋方塊中，輸入 **AKReadSourceStreamWriteSinkStream** (您在上一節中建立的政策)。
d. 選擇 AK 原ReadSourceStreamWriteSinkStream則，然後選擇 [附加原則]。

您現在已建立應用程式用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的指 step-by-step 示，請參閱 IAM 使用者指南中的建立 IAM 角色 (主控台)。

建立 Managed Service for Apache Flink 應用程式

1. 將下列 JSON 程式碼複製到名為 `create_request.json` 的檔案。以您之前建立之角色的 ARN，取代範例角色 ARN。以您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (*username*)。以您的帳戶 ID 取代服務執行角色中的範例帳戶 ID (*012345678901*)。

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-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. 以建立應用程式的上述請求，執行 CreateApplication 動作：

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

現在已建立應用程式。您會在下一個步驟中啟動應用程式。

啟動應用程式

在本節中，您會透過 StartApplication 動作來啟動應用程式。

啟動應用程式

1. 將下列 JSON 程式碼複製到名為 start_request.json 的檔案。

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

```
    }
}
```

2. 以啟動應用程式的上述請求，執行 StartApplication 動作：

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

應用程式現在正在執行。您可以在 Amazon CloudWatch 主控台上查看適用於 Apache Flink 的受管服務指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 StopApplication 動作來停止應用程式。

停止應用程式

1. 將下列 JSON 程式碼複製到名為 stop_request.json 的檔案。

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

2. 以停止應用程式的上述請求，執行 StopApplication 動作：

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

現在已停止應用程式。

新增記 CloudWatch 錄選項

您可以使用AWS CLI將 Amazon CloudWatch 日誌串流新增至您的應用程式。如需搭配應用程式使用 CloudWatch 記錄的相關資訊，請參閱the section called “設定日誌”。

更新環境屬性

在本節中，您可以使用 UpdateApplication 動作來變更應用程式的環境屬性，無需重新編譯應用程式的程式碼。在此範例中，您會變更來源和目的地串流的「區域」。

更新應用程式的環境屬性

1. 將下列 JSON 程式碼複製到名為 update_properties_request.json 的檔案。

```
{"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. 使用前述請求執行 UpdateApplication 動作以更新環境屬性：

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

更新應用程式的程式碼

需要使用新版本的程式碼套件更新應用程式的程式碼時，請使用 UpdateApplicationAWS CLI 動作。

> **Note**
>
> 若要載入具有相同檔案名稱的新版應用程式的程式碼，必須指定新的物件版本。如需關於使用 Amazon S3 物件版本的詳細資訊，請參閱啟用或停用版本控制。

若要使用 AWS CLI，請從 Amazon S3 儲存貯體刪除先前的程式碼套件，上傳新版本，然後呼叫 UpdateApplication，指定相同的 Amazon S3 儲存貯體和物件名稱，以及新的物件版本。應用程式將以新的程式碼套件重新啟動。

UpdateApplication 動作的下列範例請求會重新載入應用程式的程式碼並重新啟動應用程式。將 CurrentApplicationVersionId 更新至目前的應用程式版本。您可以使用 ListApplications 或 DescribeApplication 動作來檢查目前的應用程式版本。使用您在the section called “建立兩個 Amazon Kinesis Data Streams”一節中選擇的尾碼更新儲存貯體名稱尾碼 (*<username>*)。

```
{
    "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"
                }
            }
        }
    }
}
```

後續步驟

步驟 4：清除 AWS 資源

步驟 4：清除 AWS 資源

本節包括清除在入門教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 後續步驟

刪除 Managed Service for Apache Flink 應用程式

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis Data Streams

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇，選擇「ExampleOutputStream動作」，選擇「刪除」，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

後續步驟

步驟 5：後續步驟

步驟 5：後續步驟

現在您已建立並執行 Managed Service for Apache Flink 應用程式，請參閱下列資源，取得更進階的 Managed Service for Apache Flink 解決方案。

- Amazon Kinesis 的 AWS 串流資料解決方案：Amazon Kinesis 的 AWS 串流資料解決方案可自動設定輕鬆擷取、存放、處理和交付串流資料所需的 AWS 服務。該解決方案提供了多種解決串流資料使用案例的選項。Apache Flink 的受管理服務選項提供 end-to-end串流 ETL 範例，展示真實世界的應用程式，該應用程式會根據模擬的紐約計程車資料執行分析作業。此解決方案會設定所有必要的 AWS資源，例如 IAM 角色和政策、 CloudWatch 儀表板和 CloudWatch 警示。
- 適用於 Amazon MSK 的 AWS 串流資料解決方案：適用於 Amazon MSK 的 AWS 串流資料解決方案提供 AWS CloudFormation 範本，可讓資料流經生產者、串流儲存、取用者和目的地。
- 使用 Apache Flink 和 Apache Kafka 的點擊流實驗室：點擊流使用案例的端對端實驗室，使用 Amazon Managed Streaming for Apache Kafka 進行串流儲存，使用適用於 Apache Flink 應用程式的 Managed Service for Apache Flink 進行串流處理。
- 適用於 Apache Flink 工作坊的 Amazon 受管服務：在本研討會中，您可以建立 end-to-end 串流架構，以近乎即時的方式擷取、分析和視覺化串流資料。您著手改善紐約市一家出租車公司的運營。您可以近乎即時地分析紐約市計程車車隊的遙測資料，以最佳化其車隊運作。
- Managed Service for Apache Flink：範例：本開發人員指南的這一節提供了在 Managed Service for Apache Flink 中建立和使用應用程式的範例。其中包含範例程式碼和 step-by-step 指示，可協助您為 Apache Flink 應用程式建立受管理服務並測試結果。
- 學習 Flink：動手訓練：官方介紹性 Apache Flink 訓練課程，可協助您開始撰寫可擴展的串流 ETL、分析和事件驅動型應用程式。

> **Note**
>
> 請注意，Managed Service for Apache Flink 不支援本訓練中使用的 Apache Flink 版本 (1.12)。您可以在阿帕奇 Flink 的 Flink 管理服務中使用 Flink 1.15.2。

Amazon Managed Service in Apache Flink (資料表 API) 入門

本節將為您介紹 Managed Service for Apache Flink 和資料表 API 的基本概念。它描述了建立和測試應用程式的可用選項。此外，它還提供了相關指示，以協助您安裝完成本指南教學課程以及建立您的第一個應用程式所需要的工具。

主題

Managed Service for Apache Flink 應用程式的元件

為了處理資料，您的 Managed Service for Apache Flink 使用 Java/Apache Maven 或 Scala 應用程式來處理輸入，並使用 Apache Flink 執行時間生成輸出。

Managed Service for Apache Flink 應用程式包含以下元件：

- 執行時間屬性：您可以使用執行時間屬性來設定應用程式，無需重新編譯應用程式的程式碼。
- 資料表來源：應用程式使用來源來消耗資料。來源連接器從 Kinesis 資料串流、Amazon MSK 主題等讀取資料。如需詳細資訊，請參閱 資料表 API 來源。
- 函數：應用程式會使用一或多個函數來處理資料。函數可以轉換、富集或彙總資料。
- 接收器：應用程式透過使用接收器生成資料到外部來源。接收器連接器會將資料寫入 Kinesis 資料串流、Kinesis Data Firehose 串流、Amazon MSK 主題、Amazon S3 儲存貯體等。如需詳細資訊，請參閱 資料表 API 接收器。

建立、編譯和封裝應用程式的程式碼後，將程式碼套件上傳到 Amazon S3 儲存貯體。然後建立 Managed Service for Apache Flink 應用程式。您可以傳入程式碼套件位置，將 Amazon MSK 主題作為串流資料來源，通常是接收應用程式處理資料的串流或檔案位置。

必要條件

開始本教學課程之前，請先完成 開始使用適用於阿帕奇 Flink 的 Amazon 託管服務 (DataStream API) 中的前兩個步驟：

- 步驟 1：設定 AWS 帳戶並建立管理員使用者
- 步驟 2：設定 AWS Command Line Interface (AWS CLI)

若要開始使用，請參閱 建立應用程式。

建立並執行 Managed Service for Apache Flink 應用程式

在本練習中，您會建立 Managed Service for Apache Flink 應用程式，並將 Amazon MSK 主題作為來源，將 Amazon S3 儲存貯體作為接收器。

本節包含下列步驟：

- 建立相依資源
- 寫入範例記錄至輸入串流
- 下載並檢查 Apache Flink 串流 Java 程式碼
- 編譯應用程式的程式碼
- 上傳 Apache Flink 串流 Java 程式碼
- 建立並執行 Managed Service for Apache Flink 應用程式
- 後續步驟

建立相依資源

為本練習建立 Managed Service of Apache Flink 之前，請先建立下列相依資源：

- 以 Amazon VPC 和 Amazon MSK 叢集為基礎的虛擬私有雲端 (VPC)
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (`ka-app-code-<username>`)

建立 VPC 和 Amazon MSK 叢集

若要建立 VPC 和 Amazon MSK 叢集以從 Managed Service for Apache Flink 應用程式存取，請按照《Amazon MSK 使用入門》https://docs.aws.amazon.com/msk/latest/developerguide/gs-table.html教學課程進行操作。

完成教學課程時，請注意以下事項：

- 記錄叢集的啟動伺服器清單。您可以使用以下命令取得啟動伺服器的清單，以 MSK 叢集的 Amazon Resource Name (ARN) 取代 *ClusterArn*：

```
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"
}
```

- 按照教學課程中的步驟進行操作時，請務必在程式碼、命令和主控台項目中使用您選取的 AWS 區域。

建立 Amazon S3 儲存貯體

您可以使用主控台建立 Amazon S3 儲存貯體。如需關於建立這些資源的指示，請參閱以下主題：

- 《Amazon Simple Storage Service 使用者指南》中的如何建立 S3 儲存貯體？。透過附加登入名稱 (例如 **ka-app-code-*<username>***)，為 Amazon S3 儲存貯體提供全域唯一的名稱。

其他資源

當您建立應用程式時，Apache Flink 的受管服務會建立下列 Amazon CloudWatch 資源 (如果這些資源尚未存在)：

- 名為 `/AWS/KinesisAnalytics-java/MyApplication` 的日誌群組。
- 名為 `kinesis-analytics-log-stream` 的日誌串流。

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的 Amazon MSK 主題。

1. 連線到您在《Amazon MSK 使用入門》https://docs.aws.amazon.com/msk/latest/developerguide/gs-table.html教學課程步驟 4：建立用戶端機器中建立的用戶端執行個體。
2. 安裝 Python3 和 Kafka Python 程式庫：

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

3. 使用下列內容建立名為 `stock.py` 的檔案。以您之前記錄的啟動代理程式清單取代 `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. 在教學課程後半段，您會執行 stock.py 指令碼來傳送資料至應用程式。

```
$ python3 stock.py
```

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的 Java 應用程式程式碼可從中取得 GitHub。

下載 Java 應用程式的程式碼

1. 使用以下指令複製遠端儲存庫：

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

2. 請前往 amazon-kinesis-data-analytics-java-examples/GettingStartedTable 目錄。

請留意下列與應用程式的程式碼相關的資訊：

- 專案物件模型 (pom.xml) 檔案包含應用程式的組態和相依性資訊，包括 Managed Service for Apache Flink 程式庫。
- StreamingJob.java 檔案包含定義應用程式功能的 main 方法。
- 應用程式使用 FlinkKafkaConsumer 從 Amazon MSK 主題讀取。以下程式碼片段會建立 FlinkKafkaConsumer 物件：

```
final FlinkKafkaConsumer<StockRecord> consumer = new
 FlinkKafkaConsumer<StockRecord>(kafkaTopic, new KafkaEventDeserializationSchema(),
 kafkaProps);
```

- 您的應用程式會建立來源與目的地連接器，以使用 StreamExecutionEnvironment 和 TableEnvironment 物件來存取外部資源。
- 應用程式會使用動態應用程式屬性建立來源和接收器連接器，因此您可以指定應用程式參數 (例如 S3 儲存貯體)，無需重新編譯程式碼。

```
//read the parameters from the Managed Service for Apache Flink 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();
}
```

如需執行時間屬性的詳細資訊，請參閱執行時間屬性。

> **Note**
>
> 建置應用程式時，我們強烈建議您在與 Amazon MSK 叢集相同的區域中建立並執行 Managed Service for Apache Flink 應用程式。這是因為，Flink Kafka 連接器預設會針對低延遲環境進行最佳化。如果您需要從跨區域 Kafka 叢集取用資料，請考慮增加 `receive.buffer.byte` 的組態值，例如 2097152。
> 如需 MSK 組態的詳細資訊，請參閱自訂 MSK 組態。

編譯應用程式的程式碼

在本節中，您會使用 Apache Maven 編譯器來建立應用程式的 Java 程式碼。如需安裝 Apache Maven 和 Java 開發套件 (JDK) 的詳細資訊，請參閱 完成練習的先決條件。

編譯應用程式的程式碼

1. 請將應用程式的程式碼編譯並封裝成 JAR 檔案，以使用應用程式的程式碼。您可以使用下列兩種方式的其中之一，編譯和封裝您的程式碼：

   - 使用命令列 Maven 工具。請在包含 `pom.xml` 檔案的目錄中執行下列命令，來建立 JAR 檔案：

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

- 設定開發環境。如需詳細資訊，請參閱您的開發環境文件。

> **Note**
>
> 提供的來源程式碼依賴於 Java 11 中的程式庫。

您可以將您的套件做為 JAR 檔案上傳，或壓縮您的套件並做為 ZIP 檔案上傳。如果是使用 AWS CLI 建立應用程式，您會指定程式碼內容類型 (JAR 或 ZIP)。

2. 如果編譯時發生錯誤，請確認您的 `JAVA_HOME` 環境變數是否正確設定。

如果應用程式成功編譯，則會建立下列檔案：

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

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會建立 Amazon S3 儲存貯體並上傳您應用程式的程式碼。

上傳應用程式的程式碼

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 選擇建立儲存貯體。
3. 在儲存貯體名稱欄位中，輸入 **ka-app-code-*<username>***。新增尾碼至儲存貯體名稱，例如您的使用者名稱，使其成為全域唯一的。選擇下一步。
4. 在設定選項步驟中，保留原有設定並選擇 Next (下一步)。
5. 在設定許可步驟中，保留原有設定並選擇 Next (下一步)。
6. 選擇建立儲存貯體。
7. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
8. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `aws-kinesis-analytics-java-apps-1.0.jar` 檔案。選擇下一步。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 對於 Description (說明)，輸入 **My java test app**。
   - 針對執行時間，選擇 Apache Flink。
   - 將版本保留為 Apache Flink 版本 1.15.2（建議版本）。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - 角色：`kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

編輯 IAM 政策

編輯 IAM 政策以新增 Amazon S3 儲存貯體存取許可。

編輯 IAM 政策以新增 Amazon S3 儲存貯體許可

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。

2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。

```
{
    "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"
            ]
        }
    ]
}
```

設定應用程式

請使用下列程序設定應用程式。

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **aws-kinesis-analytics-java-apps-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在屬性下，選擇建立群組。
5. 輸入下列資料：

群組 ID	金鑰	值
FlinkApplicationProperties	**kafka-topic**	**AWSKafkaTutorialTopic**

群組 ID	金鑰	值
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. 在監控下，確保監控指標層級設為應用程式。
7. 若要CloudWatch 記錄，請選取 [啟用] 核取方塊。
8. 在虛擬私有雲端 (VPC) 區段中，選擇以 Amazon MSK 叢集為基礎的 VPC 組態。選擇AWSKafkaTutorialCluster。
9. 選擇更新。

> **Note**
>
> 當您選擇啟用 Amazon 日誌 CloudWatch 記錄時，Apache Flink 的受管服務會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`

執行應用程式

請使用下列程序執行應用程式。

執行應用程式

1. 在MyApplication頁面上，選擇 [執行]。確認動作。
2. 應用程式執行時，重新整理頁面。主控台會顯示 Application graph (應用程式圖形)。
3. 從您的 Amazon EC2 用戶端執行先前建立的 Python 指令碼，以將記錄寫入 Amazon MSK 叢集供應用程式處理：

```
$ python3 stock.py
```

停止應用程式

若要停止應用程式，請在MyApplication頁面上選擇 [停止]。確認動作。

後續步驟

清除 AWS 資源

清除 AWS 資源

本節包括清除在入門 (資料表 API) 教學課程中建立之 AWS 資源的程序。

本主題包含下列章節。

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Amazon MSK 叢集
- 刪除 VPC
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源
- 後續步驟

刪除 Managed Service for Apache Flink 應用程式

請使用下列程序刪除應用程式。

刪除應用程式

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式頁面中，選擇刪除，然後確認刪除。

刪除 Amazon MSK 叢集

若要刪除 Amazon MSK 叢集，請按照《Amazon Managed Streaming for Apache Kafka 開發人員指南》https://docs.aws.amazon.com/msk/latest/developerguide/what-is-msk.html中的步驟 8：刪除 Amazon MSK 叢集操作。

刪除 VPC

若要刪除 Amazon VPC，請執行下列動作：

- 開啟 Amazon VPC console (Amazon VPC 主控台)。
- 選擇您的 VPC。
- 對於 Actions (動作)，請選擇 Delete VPC (刪除 VPC)。

刪除 Amazon S3 物件與儲存貯體

使用下列程序來刪除 S3 物件和儲存貯體。

刪除 S3 物件與儲存貯體

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 選擇 ka-app-code- <username>桶。
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

使用下列程序刪除 IAM 資源。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

請使用下列程序刪除資 CloudWatch 源。

若要刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

後續步驟

後續步驟

後續步驟

現在您已建立並執行使用資料表 API 的 Managed Service for Apache Flink 應用程式，請參閱步驟 5：後續步驟中的開始使用適用於阿帕奇 Flink 的 Amazon 託管服務 (DataStream API)。

適用於 Python 的 Amazon Managed Service for Apache Flink 入門

本節將為您介紹使用 Python 和資料表 API 的 Managed Service for Apache Flink 的基本概念。它描述了建立和測試應用程式的可用選項。此外，它還提供了相關指示，以協助您安裝完成本指南教學課程以及建立您的第一個應用程式所需要的工具。

主題

> **Note**
>
> 如果您正在使用蘋果矽晶片的新 Mac 上開發 Python Flink 應用程式，您可能會遇到一些與 Python 相依性 PyFlink 1.15 相依性的已知問題。在這種情況下，我們建議在 Docker 中執行 Python 解譯器。如需 step-by-step 指示，請參閱蘋果矽 Mac 的 PyFlink 1.15 開發。

Pyflink - Apache 的 Python 解譯器入門 | Amazon Web Services

開始操作前，建議您觀看下列視訊：

Pyflink - Apache 的 Python 解譯器入門 | Amazon Web Services

Managed Service for Apache Flink 應用程式的元件

為了處理資料，您的 Managed Service for Apache Flink 使用 Python 應用程式來處理輸入，並使用 Apache Flink 執行時間生成輸出。

Managed Service for Apache Flink 應用程式包含以下元件：

- 執行時間屬性：您可以使用執行時間屬性來設定應用程式，無需重新編譯應用程式的程式碼。
- 資料表來源：應用程式使用來源來消耗資料。來源連接器從 Kinesis 資料串流、Amazon MSK 主題等讀取資料。如需詳細資訊，請參閱 資料表 API 來源。
- 函數：應用程式會使用一或多個函數來處理資料。函數可以轉換、富集或彙總資料。
- 接收器：應用程式透過使用接收器生成資料到外部來源。接收器連接器會將資料寫入 Kinesis 資料串流、Kinesis Data Firehose 串流、Amazon MSK 主題、Amazon S3 儲存貯體等。如需詳細資訊，請參閱 資料表 API 接收器。

建立和封裝應用程式的程式碼後，將程式碼套件上傳到 Amazon S3 儲存貯體。然後建立 Managed Service for Apache Flink 應用程式。您可以傳入程式碼套件位置，串流資料來源，通常是接收應用程式處理資料的串流或檔案位置。

必要條件

開始本教學課程之前，請先完成 開始使用適用於阿帕奇 Flink 的 Amazon 託管服務 (DataStream API) 中的前兩個步驟：

- 步驟 1：設定 AWS 帳戶並建立管理員使用者
- 步驟 2：設定 AWS Command Line Interface (AWS CLI)

若要開始使用，請參閱 建立應用程式。

建立並執行適用於 Python 應用程式的 Managed Service for Apache Flink

在本練習中，您會為 Python 建立 Managed Service for Apache Flink 應用程式，並將 Kinesis 資料串流作為來源和目的地。

本節包含下列步驟：

- 建立相依資源
- 寫入範例記錄至輸入串流
- 建立並檢查 Apache Flink 串流 Python 程式碼
- 將第三方相依項添加到 Python 應用

- 上傳 Apache Flink 串流 Python 程式碼
- 建立並執行 Managed Service for Apache Flink 應用程式
- 後續步驟

建立相依資源

為本練習建立 Managed Service of Apache Flink 之前，請先建立下列相依資源：

- 兩個 Kinesis 串流，用於輸入和輸出。
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (ka-app-code-*<username>*)

建立兩個 Kinesis 串流

在為本練習建立 Managed Service for Apache Flink 應用程式之前，請先建立兩個 Kinesis 資料串流 (ExampleInputStream 和 ExampleOutputStream)。您的應用程式會將這些串流用於應用程式來源和目的地串流。

您可以使用 Amazon Kinesis 主控台或以下 AWS CLI 命令來建立這些串流。如需主控台指示，請參閱《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。

建立資料串流 (AWS CLI)

1. 若要建立第一個串流 (ExampleInputStream)，請使用以下 Amazon Kinesis create-stream AWS CLI 命令。

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

2. 若要建立應用程式用來寫入輸出的第二個串流，請執行相同的命令，將串流名稱變更為 ExampleOutputStream。

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

建立 Amazon S3 儲存貯體

您可以使用主控台建立 Amazon S3 儲存貯體。如需關於建立這些資源的指示，請參閱以下主題：

- 《Amazon Simple Storage Service 使用者指南》中的如何建立 S3 儲存貯體？。透過附加登入名稱 (例如 **ka-app-code-*<username>***)，為 Amazon S3 儲存貯體提供全域唯一的名稱。

其他資源

當您建立應用程式時，Apache Flink 的受管服務會建立下列 Amazon CloudWatch 資源 (如果這些資源尚未存在)：

- 名為 `/AWS/KinesisAnalytics-java/MyApplication` 的日誌群組。
- 名為 `kinesis-analytics-log-stream` 的日誌串流。

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

> **Note**
>
> 本節中的 Python 指令碼會使用 AWS CLI。您必須將 AWS CLI 設定為使用您的帳戶憑證和預設區域。若要設定 AWS CLI，請輸入下列內容：
>
> ```
> aws configure
> ```

1. 使用下列內容建立名為 `stock.py` 的檔案：

```
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. 執行 stock.py 指令碼：

```
$ python stock.py
```

在完成教學課程的其餘部分時，讓指令碼保持執行狀態。

建立並檢查 Apache Flink 串流 Python 程式碼

此範例的 Python 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 請前往 amazon-kinesis-data-analytics-java-examples/python/GettingStarted 目錄。

應用程式的程式碼位於 getting_started.py 檔案中。請留意下列與應用程式的程式碼相關的資訊：

- 應用程式使用 Kinesis 資料表來源從來源串流讀取。下列程式碼片段會呼叫 create_table 函數來建立 Kinesis 資料表來源：

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

create_table 函數使用 SQL 命令來建立由串流來源支援的資料表：

```
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)
 }
```

- 應用程式會建立兩個資料表，然後將一個資料表的內容寫入另一個資料表。

```
    # 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))
```

- 此應用程式會使用 Flink - sql-connector-kinesis 檔案中的 Flink 連接器。

將第三方相依項添加到 Python 應用

使用第三方 python 套件 (例如 boto3) 時，您需要新增它們的可轉移相依項以及定位這些相依項所需的屬性。在高層次上，對於 PyPi 依賴關係，您可以複製位於 python 環境文件夾中的文件和文件`site-packages`夾，以創建如下所示的目錄結構：

```
PythonPackages
#   README.md
#   python-packages.py
#
####my_deps
       ####boto3
       #   #   session.py
       #   #   utils.py
       #   #   ...
       #
       ####botocore
       #   #   args.py
       #   #   auth.py
       #    ...
       ####mynonpypimodule
       #   #   mymodulefile1.py
       #   #   mymodulefile2.py
            ...
####lib
#    # flink-sql-connector-kinesis-1.15.2.jar
#    # ...
...
```

將 boto3 新增為第三方相依項：

1. 使用所需的相依項在本機電腦上建立一個獨立的 Python 環境（conda 或類似環境）。
2. 請注意該環境 `site_packages` 資料夾中的初始套件清單。
3. `pip-install` 您應用程式的所有必需相依項。
4. 請注意上述步驟 3 之後新增至 `site_packages` 資料夾的套件。這些是您需要包含在套件中的資料夾 (在 `my_deps` 資料夾下)，如上所示組織。這將允許您在步驟 2 和 3 之間捕獲套件的差異，以識別應用程式的正確套件相依性。
5. 提供 `my_deps/` 作為 `kinesis.analytics.flink.run.options` 屬性群組中 `pyFiles` 屬性的引數，如下述 `jarfiles` 屬性所述。Flink 還允許您使用 add_python_file 函數指定 Python 相依項，但請注意，您只需要指定一個或另一個，不必同時指定兩個。

> **Note**
>
> 您不必為資料夾 `my_deps` 命名。重要的部分是使用 `pyFiles` 或 `add_python_file` 註冊相依項。可以在如何在 PyfLink 中使用 boto3 找到一個範例。

上傳 Apache Flink 串流 Python 程式碼

在本節中，您會建立 Amazon S3 儲存貯體並上傳您應用程式的程式碼。

使用主控台上傳應用程式的程式碼：

1. 請使用您偏好的壓縮應`getting-started.py`用程式來 sql-connector-kinesis壓縮和 https://mvnrepository.com/artifact/org.apache.flink/flink 命名存檔 `myapp.zip`。如果您將外部資料夾包含在存檔中，則必須將其包含在路徑中，並將程式碼包含在組態檔案中：`GettingStarted/getting-started.py`。
2. 前往 https://console.aws.amazon.com/s3/ 開啟 Amazon S3 主控台。
3. 選擇建立儲存貯體。
4. 在儲存貯體名稱欄位中，輸入 **`ka-app-code-<username>`**。新增尾碼至儲存貯體名稱，例如您的使用者名稱，使其成為全域唯一的。選擇下一步。
5. 在設定選項步驟中，保留原有設定並選擇 Next (下一步)。
6. 在設定許可步驟中，保留原有設定並選擇 Next (下一步)。
7. 選擇建立儲存貯體。

8. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
9. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 myapp.zip 檔案。選擇下一步。
10. 您不需要變更物件的任何設定，因此請選擇上傳。

使用 AWS CLI 上傳應用程式的程式碼：

> **Note**
>
> 請勿使用 Finder (macOS) 或 Windows 檔案總管 (Windows) 中的壓縮功能來建立 myapp.zip 存檔。這麼做可能會導致應用程式的程式碼無效。

1. 請使用您偏好的壓縮應streaming-file-sink.py用程式來 sql-connector-kinesis壓縮和 https://mvnrepository.com/artifact/org.apache.flink/flink

   > **Note**
   >
   > 請勿使用 Finder (macOS) 或 Windows 檔案總管 (Windows) 中的壓縮功能來建立 myapp.zip 存檔。這麼做可能會導致應用程式的程式碼無效。

2. 使用您偏好的壓縮應用程式來壓縮getting-started.py和 https://mvnrepository.com/artifact/org.apache.flink/ flink-sql-connector-kinesis /1.15.2 檔案。命名存檔 myapp.zip。如果您將外部資料夾包含在存檔中，則必須將其包含在路徑中，並將程式碼包含在組態檔案中：GettingStarted/getting-started.py。
3. 執行以下命令：

   ```
   $ aws s3 --region aws region cp myapp.zip s3://ka-app-code-<username>
   ```

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台

2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 對於 Description (說明)，輸入 **My java test app**。
   - 針對執行時間，選擇 Apache Flink。
   - 將版本保留為 Apache Flink 版本 1.15.2（建議版本）。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：kinesis-analytics-service-*MyApplication*-*us-west-2*
> - 角色：kinesisanalytics-*MyApplication*-*us-west-2*

設定應用程式

請使用下列程序設定應用程式。

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **myapp.zip**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在屬性下，選擇新增群組。
5. 輸入下列資料：

群組 ID	金鑰	值
consumer.config.0	input.stream.name	ExampleInputStream
consumer.config.0	aws.region	us-west-2
consumer.config.0	scan.stream.initpos	LATEST

選擇儲存。

6. 在屬性下，再次選擇新增群組。
7. 輸入下列資料：

群組 ID	金鑰	值
producer.config.0	output.stream.name	ExampleOutputStream
producer.config.0	aws.region	us-west-2
producer.config.0	shard.count	1

8. 在屬性下，再次選擇新增群組。針對群組 ID，輸入 **kinesis.analytics.flink.run.options**。這個特殊的屬性群組會告訴您的應用程式在何處尋找其程式碼資源。如需詳細資訊，請參閱 指定程式碼檔案。
9. 輸入下列資料：

群組 ID	金鑰	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. 在監控下，確保監控指標層級設為應用程式。
11. 對於CloudWatch 記錄，請選擇啟用核取方塊。

12. 選擇更新。

> **Note**
>
> 當您選擇啟用 Amazon 日誌 CloudWatch 記錄時，Apache Flink 的受管服務會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`

編輯 IAM 政策

編輯 IAM 政策以新增 Amazon S3 儲存貯體存取許可。

編輯 IAM 政策以新增 Amazon S3 儲存貯體許可

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
```

```
}
```

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

停止應用程式

若要停止應用程式，請在MyApplication頁面上選擇 [停止]。確認動作。

後續步驟

清除 AWS 資源

清除 AWS 資源

本節包括清除在入門 (Python) 教學課程中建立之 AWS 資源的程序。

本主題包含下列章節。

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis Data Streams
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

請使用下列程序刪除應用程式。

刪除應用程式

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis Data Streams

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇，選擇「ExampleOutputStream動作」，選擇「刪除」，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

使用下列程序來刪除 S3 物件和儲存貯體。

刪除 S3 物件與儲存貯體

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 選擇 ka-app-code- <username>桶。
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

使用下列程序刪除 IAM 資源。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

請使用下列程序刪除資 CloudWatch 源。

若要刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

入門 (Scala)

> **Note**
>
> 從 Flink 1.15 版開始，移除了 Scala 相依性。應用程式現在可以使用任何 Scala 版本的 Java API。Flink 仍然在內部的幾個關鍵元件中使用 Scala，但不會將 Scala 公開給使用者程式碼類別加載器。因此，使用者需要將 Scala 相依項添加到他們的 jar 存檔中。
> 如需 Flink 1.15 中的 Scala 變更之詳細資訊，請參閱在 1.15 版中移除了 Scala 相依性。

在本練習中，您會為 Scala 建立 Managed Service for Apache Flink 應用程式，並將 Kinesis 資料串流作為來源和目的地。

本主題包含下列章節：

- 建立相依資源
- 寫入範例記錄至輸入串流
- 下載並檢查應用程式的程式碼
- 建立和編譯應用程式的程式碼
- 建立並執行應用程式 (主控台)
- 建立並執行應用程式 (CLI))
- 清除 AWS 資源

建立相依資源

為本練習建立 Managed Service of Apache Flink 應用程式之前，請先建立下列相依資源：

- 兩個 Kinesis 串流，用於輸入和輸出。
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (`ka-app-code-<username>`)

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需關於建立這些資源的指示，請參閱以下主題：

- 《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。為資料串流 **`ExampleInputStream`** 和 **`ExampleOutputStream`** 命名。

建立資料串流 (AWS CLI)

- 若要建立第一個串流 (ExampleInputStream)，請使用以下 Amazon Kinesis create-stream AWS CLI 命令。

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

- 若要建立應用程式用來寫入輸出的第二個串流，請執行相同的命令，將串流名稱變更為 ExampleOutputStream。

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

- 《Amazon Simple Storage Service 使用者指南》中的如何建立 S3 儲存貯體？。透過附加登入名稱 (例如 **ka-app-code-*<username>***)，為 Amazon S3 儲存貯體提供全域唯一的名稱。

其他資源

建立應用程式時，Managed Service for Apache Flink 會建立下列 Amazon CloudWatch 資源 (如果尚不存在該資源)：

- 名為 /AWS/KinesisAnalytics-java/MyApplication 的日誌群組。
- 名為 kinesis-analytics-log-stream 的日誌串流。

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

> **Note**
>
> 本節中的 Python 指令碼會使用AWS CLI。您必須將 AWS CLI 設定為使用您的帳戶憑證和預設區域。若要設定 AWS CLI，請輸入下列內容：
>
> ```
> aws configure
> ```

1. 使用下列內容建立名為 stock.py 的檔案：

```
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. 執行 stock.py 指令碼：

```
$ python stock.py
```

在完成教學課程的其餘部分時，讓指令碼保持執行狀態。

下載並檢查應用程式的程式碼

此範例的 Python 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，執行下列操作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 請前往 `amazon-kinesis-data-analytics-java-examples/scala/GettingStarted` 目錄。

請留意下列與應用程式的程式碼相關的資訊：

- `build.sbt` 檔案包含應用程式的組態和相依性資訊，包括 Managed Service for Apache Flink 程式庫。
- `BasicStreamingJob.scala` 檔案包含定義應用程式功能的主要方法。
- 應用程式使用 Kinesis 來源從來源串流讀取。以下程式碼片段會建立 Kinesis 來源：

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

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

應用程式也會使用 Kinesis 接收器寫入結果串流。以下程式碼片段會建立 Kinesis 目的地：

```
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
}
```

- 應用程式會建立來源與目的地連接器，以使用 StreamExecutionEnvironment 物件來存取外部資源。
- 應用程式會使用動態應用程式屬性來建立來源與目的地連接器。會讀取執行時間應用程式的屬性，來設定連接器。如需執行時間屬性的詳細資訊，請參閱執行時間屬性。

建立和編譯應用程式的程式碼

在本節中，您會編譯應用程式的程式碼，並將其上傳至在建立相依資源一節建立的 Amazon S3 儲存貯體。

編譯應用程式的程式碼

在本節中，您將使用 SBT 建置工具來建置應用程式的 Scala 程式碼。若要安裝 SBT，請參閱使用 cs 安裝程式安裝 sbt。您還需要安裝 Java 開發套件 (JDK)。請參閱完成練習的先決條件。

1. 請將應用程式的程式碼編譯並封裝成 JAR 檔案，以使用應用程式的程式碼。您可以使用 SBT 編譯和封裝程式碼：

   ```
   sbt assembly
   ```

2. 如果應用程式成功編譯，則會建立下列檔案：

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

上傳 Apache Flink 串流 Scala 程式碼

在本節中，您會建立 Amazon S3 儲存貯體並上傳您應用程式的程式碼。

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇建立儲存貯體。
3. 在儲存貯體名稱欄位中，輸入 ka-app-code-<username>。新增尾碼至儲存貯體名稱，例如您的使用者名稱，使其成為全域唯一的。選擇下一步。

4. 在設定選項中，保留原有設定並選擇下一步。
5. 在設定許可步驟中，保留原有設定並選擇下一步。
6. 選擇 建立儲存貯體 。
7. 選擇 ka-app-code-<username> 儲存貯體，然後選擇上傳。
8. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 getting-started-scala-1.0.jar 檔案。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行應用程式 (主控台)

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 對於 Description (說明)，輸入 **My scala test app**。
   - 針對執行時間，選擇 Apache Flink。
   - 將版本保留為 Apache Flink 版本 1.15.2 (建議版本)。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：

- 政策：kinesis-analytics-service-*MyApplication*-*us-west-2*
- 角色：kinesisanalytics-*MyApplication*-*us-west-2*

設定應用程式

請使用下列程序設定應用程式。

設定應用程式

1. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **getting-started-scala-1.0.jar.**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在屬性下，選擇新增群組。
5. 輸入下列：

群組 ID	索引鍵	值
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

   選擇儲存。
6. 在屬性下，再次選擇新增群組。
7. 輸入下列：

群組 ID	索引鍵	值
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. 在監控下，確保監控指標層級設為應用程式。
9. 針對 CloudWatch 記錄，選取啟用核取方塊。
10. 選擇更新。

> **Note**
>
> 當您選擇啟用 Amazon CloudWatch 日誌時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`

編輯 IAM 政策

編輯 IAM 政策以新增 Amazon S3 儲存貯體存取許可。

編輯 IAM 政策以新增 Amazon S3 儲存貯體許可

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面上，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

停止應用程式

若要停止應用程式，請在 MyApplication 頁面上選擇停止。確認動作。

建立並執行應用程式 (CLI))

在本節中，您可以使用 AWS Command Line Interface 建立和執行 Managed Service for Apache Flink 應用程式。使用 KinesisIntictictics v2 AWS CLI 命令建立 Managed Service for Apache Flink 應用程式並與之互動。

建立許可政策

> **Note**
>
> 您必須為應用程式建立許可政策和角色。如果您未建立這些 IAM 資源，應用程式將無法存取其資料和日誌串流。

您會先建立具有兩條陳述式的許可政策：一條陳述式授予來源串流上讀取動作的許可，另一條則是授予目的地串流上寫入動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當

Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

使用以下程式碼來建立 AKReadSourceStreamWriteSinkStream 許可政策。以您用於建立 Amazon S3 儲存貯體 (以儲存應用程式的程式碼) 的使用者名稱來取代 **username**。以您的帳戶 ID 取代 Amazon Resource Name (ARN) (**012345678901**) 中的帳戶 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"
        }
    ]
}
```

如需建立許可政策的逐步指示，請參閱《IAM 使用者指南》中的教學課程：建立和附接您的第一個客戶管理政策。

建立 IAM 政策

在本節中，您會建立 Managed Service for Apache Flink 應用程式可以擔任的 IAM 角色，以便讀取來源串流與寫入目的地串流。

Managed Service for Apache Flink 沒有許可，無法存取串流。您可以透過 IAM 角色來授予這些許可。各 IAM 角色都有連接兩項政策。信任政策會授予擔任角色的 Managed Service for Apache Flink 許可，而許可政策決定了 Managed Service for Apache Flink 在擔任角色後可以執行的作業。

您會將在上一節中建立的許可政策連接至此角色。

若要建立一個 IAM 角色

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽窗格中，選擇角色，然後選擇建立角色。
3. 在選取可信身分類型下，選擇 AWS 服務。
4. 在 Choose the service that will use this role (選擇將使用此角色的服務) 下，選擇 Kinesis。
5. 在選取您的使用案例下，選擇 Managed Service for Apache Flink。
6. 選擇 Next: Permissions (下一步：許可)。
7. 在 Attach permissions policies (連接許可政策) 頁面上，選擇 Next: Review (下一步：檢閱)。您會在建立角色後連接許可政策。
8. 在建立角色頁面，輸入 **MF-stream-rw-role** 作為角色名稱。選擇建立角色。

   現在您已建立新的 IAM 角色，名為 `MF-stream-rw-role`。您接著會更新角色的信任和許可政策

9. 將 許可政策連接到角色。

   > **Note**
   >
   > 在此練習中，Managed Service for Apache Flink 擔任從 Kinesis 資料串流 (來源) 讀取資料並將輸出寫入另一個 Kinesis 資料串流的角色。因此您會連接在上一個步驟建立許可政策中建立的政策。

   a. 在摘要頁面上，選擇許可標籤。
   b. 選擇 Attach Policies (連接政策)。
   c. 在搜尋方塊中，輸入 **AKReadSourceStreamWriteSinkStream** (您在上一節中建立的政策)。
   d. 選擇 `AKReadSourceStreamWriteSinkStream` 政策，然後選擇連接政策。

您現在已建立應用程式用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的逐步說明，請參閱《IAM 使用者指南》中的建立 IAM 角色 (主控台)。

建立應用程式

將下列 JSON 程式碼複製到名為 `create_request.json` 的檔案。以您之前建立之角色的 ARN，取代範例角色 ARN。以您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (username)。以您的帳戶 ID 取代服務執行角色中的範例帳戶 ID (012345678901)。

```
{
    "ApplicationName": "getting_started",
    "ApplicationDescription": "Scala getting started application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-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"
      }
   ]
}
```

使用下列請求執行 CreateApplication 以建立應用程式：

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

現在已建立應用程式。您會在下一個步驟中啟動應用程式。

啟動應用程式

在本節中，您會使用 StartApplication 動作來啟動應用程式。

啟動應用程式

1. 將下列 JSON 程式碼複製到名為 `start_request.json` 的檔案。

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

2. 以啟動應用程式的上述請求，執行 `StartApplication` 動作：

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

應用程式現在正在執行。您可以在 Amazon CloudWatch 主控台上查看 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用 StopApplication 動作來停止應用程式。

停止應用程式

1. 將下列 JSON 程式碼複製到名為 stop_request.json 的檔案。

```
{
   "ApplicationName": "s3_sink"
}
```

2. 使用前述請求執行 StopApplication 動作以停止應用程式：

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

現在已停止應用程式。

新增 CloudWatch 記錄選項

您可以使用 AWS CLI 將 Amazon CloudWatch 日誌串流新增至應用程式。如需關於搭配應用程式使用 CloudWatch Logs 的詳細資訊，請參閱設定應用程式記錄。

更新環境屬性

在本節中，您可以使用 UpdateApplication 動作來變更應用程式的環境屬性，無需重新編譯應用程式的程式碼。在此範例中，您會變更來源和目的地串流的「區域」。

更新應用程式的環境屬性

1. 將下列 JSON 程式碼複製到名為 update_properties_request.json 的檔案。

```
{
      "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. 使用前述請求執行 UpdateApplication 動作以更新環境屬性：

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

更新應用程式的程式碼

需要使用新版本的程式碼套件更新應用程式的程式碼時，請使用 UpdateApplication CLI 動作。

> **Note**
>
> 若要載入具有相同檔案名稱的新版應用程式的程式碼，必須指定新的物件版本。如需關於使用 Amazon S3 物件版本的詳細資訊，請參閱啟用或停用版本控制。

若要使用 AWS CLI，請從 Amazon S3 儲存貯體刪除先前的程式碼套件，上傳新版本，然後呼叫 UpdateApplication，指定相同的 Amazon S3 儲存貯體和物件名稱，以及新的物件版本。應用程式將以新的程式碼套件重新啟動。

UpdateApplication 動作的下列範例請求會重新載入應用程式的程式碼並重新啟動應用程式。將 CurrentApplicationVersionId 更新至目前的應用程式版本。您可以使用 ListApplications 或 DescribeApplication 動作來檢查目前的應用程式版本。使用您在建立相依資源一節中選擇的尾碼更新儲存貯體名稱尾碼 (<username>)。

```
{{
    "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"
                }
            }
        }
    }
}
```

清除 AWS 資源

本節包括清除在「輪轉視窗」教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis Data Streams
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service in Apache Flink 面板中，選擇我的應用程式。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis Data Streams

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在「Kinesis Data Streams」面板中，選擇 ExampleInputStream。
3. 在 ExampleInputStream 頁面中，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code-***<username>*** 儲存貯體。
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service-MyApplication-us-west-2 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

建立使用 Apache Beam 的 Managed Service in Apache Flink 應用程式

您可以將 Apache Beam 架構與 Managed Service for Apache Flink 搭配使用來處理串流資料。使用 Apache Beam 的 Managed Service for Apache Flink 應用程式使用 Apache Flink 執行器來執行 Beam 管道。

如需關於如何在 Managed Service for Apache Flink 中使用 Apache Beam 的教學課程，請參閱搭配使用 CloudFormation 與 Managed Service for Apache Flink。

本主題包含下列章節：

- 將 Apache Beam 與 Managed Service for Apache Flink 搭配使用
- Beam 功能
- 使用 Apache Beam 建立應用程式

將 Apache Beam 與 Managed Service for Apache Flink 搭配使用

請注意下列有關將 Apache Flink 執行器與 Managed Service for Apache Flink 搭配使用的相關資訊：

- 在 Managed Service in Apache Flink 主控台中無法檢視 Apache Beam 指標。
- 只有使用 Apache Flink 1.8 及以上版本的 Managed Service for Apache Flink 應用程式才支援 Apache Beam。使用 Apache Flink 1.6 版的 Managed Service for Apache Flink 應用程式不支援 Apache Beam。

Beam 功能

Managed Service for Apache Flink 與 Apache Flink 執行器支援相同的 Apache Beam 功能。如需 Apache Flink 執行器所支援功能的相關資訊，請參閱 Beam 相容性矩陣。

建議您在 Managed Service for Apache Flink 服務中測試 Apache Flink 應用程式，以確認我們是否支援您的應用程式所需的全部功能。

使用 Apache Beam 建立應用程式

在本練習中，您將使用 Apache Beam 建立可轉換資料的 Managed Service for Apache Flink 應用程式。Apache Beam 是用於處理串流資料的程式設計模型。如需將 Apache Beam 與 Managed Service for Apache Flink 搭配使用的資訊，請參閱使用 Apache Beam。

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成 開始使DataStream 用 (API) 練習。

本主題包含下列章節：

建立相依資源

為本練習建立 Managed Service of Apache Flink 應用程式之前，請先建立下列相依資源：

- 兩個 Kinesis 資料串流 (`ExampleInputStream` 和 `ExampleOutputStream`)
- 用來儲存應用程式程式碼 (`ka-app-code-<username>`) 的 Amazon S3 儲存貯體

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需關於建立這些資源的指示，請參閱以下主題：

- 《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。為資料串流 **`ExampleInputStream`** 和 **`ExampleOutputStream`** 命名。
- 《Amazon Simple Storage Service 使用者指南》中的如何建立 S3 儲存貯體？。透過附加登入名稱 (例如 **`ka-app-code-<username>`**)，為 Amazon S3 儲存貯體提供全域唯一的名稱。

寫入範例記錄至輸入串流

在本節中，您會使用 Python 指令碼將隨機字串寫入串流供應用程式處理。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

1. 使用下列內容建立名為 `ping.py` 的檔案：

```
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. 執行 `ping.py` 指令碼：

```
$ python ping.py
```

   在完成教學課程的其餘部分時，讓指令碼保持執行狀態。

下載並檢查應用程式的程式碼

此範例的 Java 應用程式的程式碼可從 GitHub 下載。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 請前往 amazon-kinesis-data-analytics-java-examples/Beam 目錄。

應用程式的程式碼位於 BasicBeamStreamingJob.java 檔案中。請留意下列與應用程式的程式碼相關的資訊：

- 該應用程序使用 Apache Beam Pardo，透過調用稱為 PingPongFn 的自定義轉換函數來處理傳入的記錄。

  調用 PingPongFn 函數的代碼如下：

```
.apply("Pong transform",
    ParDo.of(new PingPongFn())
```

- 使用 Apache Beam 的 Managed Service for Apache Flink 應用程式需要下列元件。如果您未在 pom.xml 中包含這些元件和版本，應用程式會從環境相依性載入不正確的版本，而且由於版本不符合，應用程式會在執行時間損毀。

```
<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 轉換函數會將輸入資料傳遞到輸出串流，除非輸入資料是 ping，在這種情況下，它發出字串 pong\n 到輸出串流。

  轉換函數的程式碼如下：

```
    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());
        }
    }
}
```

編譯應用程式的程式碼

若要編譯應用程式，執行下列動作：

1. 如果尚未安裝 Java 和 Maven，請安裝。如需詳細資訊，請參閱開始使DataStream 用 (API)教學課程中的必要條件。
2. 使用下列命令編譯應用程式：

```
mvn package -Dflink.version=1.15.3 -Dflink.version.minor=1.8
```

> **Note**
>
> 提供的來源程式碼依賴於 Java 11 中的程式庫。

編譯應用程式會建立應用程式 JAR 檔案 (`target/basic-beam-app-1.0.jar`)。

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在建立相依資源一節建立的 Amazon S3 儲存貯體。

1. 在 Amazon S3 主控台中，選擇 ka-app-code-***<username>*** 儲存貯體，並選擇上傳。
2. 在選取檔案步驟中，選擇新增檔案。導覽至您在上一步驟中建立的 `basic-beam-app-1.0.jar` 檔案。
3. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 在以下網址開啟 Managed Service of Apache Flink 主控台：https://console.aws.amazon.com/flink
2. 在 Managed Service in Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 針對執行時間，選擇 Apache Flink。

     > **Note**
     >
     > Managed Service for Apache Flink 使用 Apache Flink 1.15.2 版。

   - 將版本下拉式清單保留為 Apache Flink 1.15.2 版 (建議版本)。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：`kinesis-analytics-service-MyApplication-us-west-2`
> - 角色：`kinesis-analytics-MyApplication-us-west-2`

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。

4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

設定應用程式

1. 在我的應用程式頁面，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **basic-beam-app-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 輸入下列：

群組 ID	索引鍵	值
BeamApplicationProperties	**InputStreamName**	**ExampleInputStream**
BeamApplicationProperties	**OutputStreamName**	**ExampleOutputStream**
BeamApplicationProperties	**AwsRegion**	**us-west-2**

5. 在監控下，確保監控指標層級設為應用程式。

6. 針對 CloudWatch 記錄，選取啟用核取方塊。
7. 選擇更新。

> **Note**
>
> 當您選擇啟用 CloudWatch 記錄時，Managed Service for Apache Flink 便會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`
>
> 此日誌串流用於監視應用程式。這與應用程式用來傳送結果的日誌串流不同。

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 任務，即可檢視 Flink 任務圖表。

您可以在 CloudWatch 主控台上查看 Managed Service for Apache Flink 指標，以確認應用程式是否正常運作。

清除 AWS 資源

本節包括清除在「輪轉視窗」教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis Data Streams
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 在以下網址開啟 Managed Service of Apache Flink 主控台：https://console.aws.amazon.com/flink

2. 在 Managed Service in Apache Flink 面板中，選擇我的應用程式。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis Data Streams

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在 Kinesis Data Streams 面板中，選擇 ExampleInputStream。
3. 在 ExampleInputStream 頁面中，選擇刪除 Kinesis 串流，然後確認刪除。
4. 在 Kinesis 串流頁面，依序選擇 ExampleOutputStream、動作和刪除，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code-***<username>*** 儲存貯體。
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service-MyApplication-us-west-2 政策。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇 kinesis-analytics-MyApplication-us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除 CloudWatch 資源

1. 在以下網址開啟 CloudWatch 主控台：https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/kinesis-analytics/MyApplication 日誌群組。
4. 選擇刪除日誌群組，然後確認刪除。

後續步驟

現在您已建立並執行使用 Apache Beam 轉換資料的基本 Managed Service for Apache Flink 應用程式，請參閱下列應用程式，取得更進階 Managed Service for Apache Flink 解決方案的範例。

- Beam 用於 Managed Service for Apache Flink 串流研討會：在此研討會中，我們將探索一個端對端範例，將批次和串流方面結合在一個統一的 Apache Beam 管道中。

培訓研討會、實驗室和解決方案實作

下列 end-to-end 範例示範適用於 Apache Flink 解決方案的進階受管理服務。

主題

- 在本機開發 Apache Flink 應用程式，然後再將其部署到 Managed Service for Apache Flink
- 使用 Managed Service for Apache Flink Studio 進行事件偵測
- Amazon Kinesis 的 AWS 串流資料解決方案
- 使用 Apache Flink 和 Apache Kafka 的點擊流實驗室
- 使用 Application Auto Scaling 進行自訂擴展
- Amazon CloudWatch 儀表
- Amazon Kinesis 的 AWS 串流資料解決方案
- 阿帕奇 Flink 解決方案的更多託管服務 GitHub

在本機開發 Apache Flink 應用程式，然後再將其部署到 Managed Service for Apache Flink

本研討會將向您展示設定並開始在本機開發 Apache Flink 應用程式的基本概念，以實現部署至 Managed Service for Apache Flink 的長期目標。

可以在這裡找到解決方案：《Apache Flink 本機開發入門指南》https://catalog.us-east-1.prod.workshops.aws/workshops/429cec9e-3222-4943-82f7-1f45c45ed99a/en-US

使用 Managed Service for Apache Flink Studio 進行事件偵測

本研討會介紹如何使用 Managed Service for Apache Flink Studio 進行事件偵測，並將其部署為 Managed Service for Apache Flink 應用程式

可以在這裡找到解決方案：使用 Managed Service for Apache Flink 進行事件偵測

Amazon Kinesis 的 AWS 串流資料解決方案

Amazon Kinesis 的 AWS 串流資料解決方案可自動設定輕鬆擷取、存放、處理和交付串流資料所需的 AWS 服務。該解決方案提供了多種解決串流資料使用案例的選項。Apache Flink 的受管理服務選項提

供 end-to-end 串流 ETL 範例，展示真實世界的應用程式，該應用程式會在模擬紐約計程車資料上執行分析作業。

每個解決方案 包含下列元件：

- 可部署完整範例的 AWS CloudFormation 套件。
- 顯示應用程式指標的 CloudWatch 儀表板。
- CloudWatch 最相關的應用程序指標的警報。
- 所有必要的 IAM 角色和政策。

可以在這裡找到解決方案：適用於 Amazon Kinesis 的串流資料解決方案

使用 Apache Flink 和 Apache Kafka 的點擊流實驗室

點擊流使用案例的端對端實驗室，使用 Amazon Managed Streaming for Apache Kafka 進行串流儲存，使用適用於 Apache Flink 應用程式的 Managed Service for Apache Flink 進行串流處理。

可以在這裡找到解決方案：點擊流實驗室

使用 Application Auto Scaling 進行自訂擴展

此範例可協助使用者使用應用程式自動擴展應用程式自動調整 Apache Flink 應用程式的受管理服務 這可讓使用者設定自訂擴展政策和自訂擴展屬性。

該解決方案可以在這裡找到：

- Apache Flink 應用程式自動調度資源的託管服務
- 排程擴展

如需有關可執行自訂擴展的詳細資訊，請參閱為 Apache Flink 的 Amazon 受管服務啟用指標式和排程擴展。

Amazon CloudWatch 儀表

用於監視 Apache Flink 應用程式的受管理服務的範例 CloudWatch 儀表板。該範例儀表板還包含示範應用程式，可協助展示儀表板的功能。

該解決方案可以在這裡找到：Managed Service for Apache Flink 指標儀表板

Amazon Kinesis 的 AWS 串流資料解決方案

適用於 Amazon MSK 的 AWS 串流資料解決方案提供了 AWS CloudFormation 範本，可讓資料流經生產者、串流儲存體、取用者和目的地。

可以在這裡找到解決方案：AWS適用於 Amazon Kinesis 的串流資料解決方案

阿帕奇 Flink 解決方案的更多託管服務 GitHub

下列 end-to-end 範例示範適用於 Apache Flink 解決方案的進階受管理服務，可在上 GitHub取得：

- Amazon Managed Service for Apache Flink — 基準參考工具
- 快照管理器 — Amazon Managed Service for Apache Flink
- 基於 Apache Flink 和 Amazon Managed Service for Apache Flink 的串流 ETL
- 對客戶意見反饋進行即時情緒分析

公用程式

下列公用程式可讓您更輕鬆地使用 Managed Service for Apache Flink 服務：

主題

快照管理員

對於 Flink 應用程式，定期觸發儲存點/快照是一種最佳實務，可以實現更順暢的故障復原。快照管理員可自動執行此任務並提供下列優點：

- 建立執行中 Managed Service for Apache Flink 應用程式的新快照
- 取得應用程式快照的計數
- 檢查計數是否超過所需的快照數
- 刪除早於所需數目的舊快照

如需範例，請參閱快照管理員。

基準測試

Managed Service for Apache Flink 基準測試公用程式可協助進行 Managed Service for Apache Flink 應用程式的容量規劃、整合測試和基準測試。

有關示例，請參閱基準測試

Managed Service for Apache Flink：範例

本節提供在 Managed Service for Apache Flink 中建立及使用應用程式的範例。其中包含範例程式碼和 step-by-step 指示，可協助您為 Apache Flink 應用程式建立受管理服務並測試結果。

在探索這些範例之前，建議先檢閱以下內容：

- 運作方式
- 開始使DataStream 用 (API)

> **Note**
>
> 這些範例假設您使用美國西部 (奧勒岡) 區域 (us-west-2)。如果您使其他區域，請相應地更新應用程式的程式碼、命令和 IAM 角色。

主題

- DataStream API 範例
- Python 範例
- Scala 範例

DataStream API 範例

下列範例示範如何使用 Apache Flink DataStream API 建立應用程式。

主題

- 範例：輪轉視窗
- 範例：滑動視窗
- 範例：寫入 Amazon S3 儲存貯體
- 教學課程：使用 Managed Service for Apache Flink 應用程式將 MSK 叢集中某個主題的資料複製到 VPC 中的另一個主題
- 範例：在 Kinesis 資料串流中使用 EFO 取用者
- 範例：寫入 Kines is Data Firehose

- 範例：從不同帳戶的 Kinesis 串流中讀取
- 教學課程：搭配 Amazon MSK 使用自訂信任存放區

範例：輪轉視窗

在本練習中，您將使用輪轉視窗建立可彙總資料的 Managed Service for Apache Flink 應用程式。彙總在 Flink 中預設為啟用。可使用下列命令將其停用：

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

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成 開始使DataStream 用 (API) 練習。

本主題包含下列章節：

- 建立相依資源
- 寫入範例記錄至輸入串流
- 下載並檢查應用程式的程式碼
- 編譯應用程式的程式碼
- 上傳 Apache Flink 串流 Java 程式碼
- 建立並執行 Managed Service for Apache Flink 應用程式
- 清除 AWS 資源

建立相依資源

在為本練習建立 Managed Service for Apache Flink 應用程式之前，先建立下列相依資源：

- 兩個 Kinesis 資料串流 (`ExampleInputStream` 和 `ExampleOutputStream`)
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (`ka-app-code-<username>`)

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需關於建立這些資源的指示，請參閱以下主題：

- 《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。為資料串流 **ExampleInputStream** 和 **ExampleOutputStream** 命名。
- 《Amazon Simple Storage Service 使用者指南》中的如何建立 S3 儲存貯體？。透過附加登入名稱 (例如 **ka-app-code-*<username>***)，為 Amazon S3 儲存貯體提供全域唯一的名稱。

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

1. 使用下列內容建立名為 `stock.py` 的檔案：

```
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. 執行 stock.py 指令碼：

```
$ python stock.py
```

在完成教學課程的其餘部分時，讓指令碼保持執行狀態。

下載並檢查應用程式的程式碼

此範例的 Java 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 請前往 amazon-kinesis-data-analytics-java-examples/TumblingWindow 目錄。

應用程式的程式碼位於 TumblingWindowStreamingJob.java 檔案中。請留意下列與應用程式的程式碼相關的資訊：

- 應用程式使用 Kinesis 來源從來源串流讀取。以下程式碼片段會建立 Kinesis 來源：

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

- 新增以下 import 陳述式：

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

- 該應用程式使用 timeWindow 運算子在 5 秒的輪轉視窗內尋找每個股票程式碼的值計數。下列程式碼會建立運算子，並將彙總的資料傳送至新的 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());
```

編譯應用程式的程式碼

若要編譯應用程式，請執行下列動作：

1. 如果尚未安裝 Java 和 Maven，請安裝。如需詳細資訊，請參閱開始使DataStream 用 (API)教學課程中的必要條件。
2. 使用下列命令編譯應用程式：

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

> **Note**
>
> 提供的來源程式碼依賴於 Java 11 中的程式庫。

編譯應用程式會建立應用程式 JAR 檔案 (`target/aws-kinesis-analytics-java-apps-1.0.jar`)。

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在建立相依資源一節建立的 Amazon S3 儲存貯體。

1. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
2. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `aws-kinesis-analytics-java-apps-1.0.jar` 檔案。
3. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 針對執行時間，選擇 Apache Flink。

     > **Note**
     >
     > Managed Service for Apache Flink 使用 Apache Flink 1.15.2 版。

   - 將版本下拉式清單保留為 Apache Flink 1.15.2 版 (建議版本)。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：`kinesis-analytics-service-MyApplication-us-west-2`
> - 角色：`kinesisanalytics-MyApplication-us-west-2`

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。

2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **aws-kinesis-analytics-java-apps-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在監控下，確保監控指標層級設為應用程式。
5. 對於CloudWatch 記錄，請選取啟用核取方塊。
6. 選擇更新。

> **Note**
>
> 當您選擇啟用 CloudWatch 記錄時，Apache Flink 的受管理服務會為您建立記錄群組和記錄資料流。這些資源的名稱如下所示：

- 日誌群組：`/aws/kinesis-analytics/MyApplication`
- 日誌串流：`kinesis-analytics-log-stream`

此日誌串流用於監控應用程式。這與應用程式用來傳送結果的日誌串流不同。

執行應用程式

1. 在MyApplication頁面上，選擇 [執行]。保持選取不使用快照執行選項，然後確認動作。
2. 應用程式執行時，重新整理頁面。主控台會顯示 Application graph (應用程式圖形)。

您可以在 CloudWatch 主控台上檢查 Apache Flink 的受管理服務量度，以確認應用程式是否正常運作。

清除 AWS 資源

本節包括清除在輪轉視窗教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis 資料串流
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis 資料串流

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。

2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇 ExampleOutputStream，選擇動作，選擇刪除，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：滑動視窗

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成 開始使DataStream 用 (API) 練習。

本主題包含下列章節：

建立相依資源

在為本練習建立 Managed Service for Apache Flink 應用程式之前，先建立下列相依資源：

- 兩個 Kinesis 資料串流 (`ExampleInputStream` 和 `ExampleOutputStream`)
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (`ka-app-code-<username>`)

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需關於建立這些資源的指示，請參閱以下主題：

- 《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。為資料串流 **`ExampleInputStream`** 和 **`ExampleOutputStream`** 命名。
- 《Amazon Simple Storage Service 使用者指南》中的如何建立 S3 儲存貯體？。透過附加登入名稱 (例如 **`ka-app-code-<username>`**)，為 Amazon S3 儲存貯體提供全域唯一的名稱。

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

1. 使用下列內容建立名為 `stock.py` 的檔案：

```
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. 執行 stock.py 指令碼：

```
$ python stock.py
```

在完成教學課程的其餘部分時，讓指令碼保持執行狀態。

下載並檢查應用程式的程式碼

此範例的 Java 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。

2. 使用以下指令複製遠端儲存庫：

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

3. 請前往 amazon-kinesis-data-analytics-java-examples/SlidingWindow 目錄。

應用程式的程式碼位於 SlidingWindowStreamingJobWithParallelism.java 檔案中。請留意下列與應用程式的程式碼相關的資訊：

- 應用程式使用 Kinesis 來源從來源串流讀取。以下程式碼片段會建立 Kinesis 來源：

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

- 該應用程序使用 timeWindow 運算子在按 5 秒滑動的 10 秒視窗內尋找每個股票代碼的最小值。下列程式碼會建立運算子，並將彙總的資料傳送至新的 Kinesis Data Streams 接收器：
- 新增以下 import 陳述式：

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

- 該應用程式使用 timeWindow 運算子在 5 秒的輪轉視窗內尋找每個股票程式碼的值計數。下列程式碼會建立運算子，並將彙總的資料傳送至新的 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());
```

編譯應用程式的程式碼

若要編譯應用程式，請執行下列動作：

1. 如果尚未安裝 Java 和 Maven，請安裝。如需詳細資訊，請參閱開始使DataStream 用 (API)教學課程中的必要條件。

2. 使用下列命令編譯應用程式：

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

> **Note**
>
> 提供的來源程式碼依賴於 Java 11 中的程式庫。

編譯應用程式會建立應用程式 JAR 檔案 (`target/aws-kinesis-analytics-java-apps-1.0.jar`)。

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在建立相依資源一節建立的 Amazon S3 儲存貯體。

1. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇 [上傳]。
2. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `aws-kinesis-analytics-java-apps-1.0.jar` 檔案。
3. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **`MyApplication`**。
   - 針對執行時間，選擇 Apache Flink。
   - 將版本下拉式清單保留為 Apache Flink 1.15.2 版 (建議版本)。
4. 對於存取許可，選擇建立/更新 IAM 角色 **`kinesis-analytics-MyApplication-us-west-2`**。

5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：kinesis-analytics-service-*MyApplication*-*us-west-2*
> - 角色：kinesisanalytics-*MyApplication*-*us-west-2*

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

設定應用程式

1. 在MyApplication頁面上，選擇設定。

2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **aws-kinesis-analytics-java-apps-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在監控下，確保監控指標層級設為應用程式。
5. 對於CloudWatch 記錄，請選取啟用核取方塊。
6. 選擇更新。

> **Note**
>
> 當您選擇啟用 Amazon 日誌 CloudWatch 記錄時，Apache Flink 的受管服務會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`
>
> 此日誌串流用於監控應用程式。這與應用程式用來傳送結果的日誌串流不同。

設定應用程式平行處理

此應用程式範例使用任務的平行執行。下列應用程式的程式碼會設定 `min` 運算子的平行處理層級：

```
.setParallelism(3) // Set parallelism for the min operator
```

應用程式平行處理層級不能大於佈建的平行處理層級 (預設值為 1)。若要增加應用程式的平行處理層級，請使用下列 AWS CLI 動作：

```
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動作擷取目前的應用程式版本 ID。

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

您可以在 CloudWatch 主控台上檢查 Apache Flink 的受管理服務量度，以確認應用程式是否正常運作。

清除 AWS 資源

本節包括清除在「滑動視窗」教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis 資料串流
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis 資料串流

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇 ExampleOutputStream，選擇動作，選擇刪除，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。

2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：寫入 Amazon S3 儲存貯體

在本練習中，您會建立 Managed Service for Apache Flink 應用程式，並將 Kinesis 資料串流作為來源，將 Amazon S3 儲存貯體作為接收器。使用接收器，您就可以驗證 Amazon S3 主控台的應用程式輸出。

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成 開始使DataStream 用 (API) 練習。

本主題包含下列章節：

- 建立相依資源
- 寫入範例記錄至輸入串流
- 下載並檢查應用程式的程式碼
- 修改應用程式的程式碼
- 編譯應用程式的程式碼
- 上傳 Apache Flink 串流 Java 程式碼
- 建立並執行 Managed Service for Apache Flink 應用程式
- 驗證應用程式輸出
- 選用：自訂來源和接收器
- 清除 AWS 資源

建立相依資源

為本練習建立 Managed Service of Apache Flink 之前，請先建立下列相依資源：

- Kinesis 資料串流 (`ExampleInputStream`)。
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (`ka-app-code-<username>`)

> **Note**
>
> Managed Service for Apache Flink 無法在其自身啟用伺服器端加密的情況下將資料寫入 Amazon S3。

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需關於建立這些資源的指示，請參閱以下主題：

- 《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。為資料串流 **`ExampleInputStream`** 命名。
- 《Amazon Simple Storage Service 使用者指南》中的如何建立 S3 儲存貯體？。透過附加登入名稱 (例如 **`ka-app-code-<username>`**)，為 Amazon S3 儲存貯體提供全域唯一的名稱。在 Amazon S3 儲存貯體中建立兩個資料夾 (**`code`** 和 **`data`**)。

如果下列 CloudWatch 資源尚未存在，應用程式會建立下列資源：

- 名為 /AWS/KinesisAnalytics-java/MyApplication 的日誌群組。
- 名為 kinesis-analytics-log-stream 的日誌串流。

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

1. 使用下列內容建立名為 stock.py 的檔案：

```
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. 執行 stock.py 指令碼：

```
$ python stock.py
```

在完成教學課程的其餘部分時，讓指令碼保持執行狀態。

下載並檢查應用程式的程式碼

此範例的 Java 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 請前往 amazon-kinesis-data-analytics-java-examples/S3Sink 目錄。

應用程式的程式碼位於 S3StreamingSinkJob.java 檔案中。請留意下列與應用程式的程式碼相關的資訊：

- 應用程式使用 Kinesis 來源從來源串流讀取。以下程式碼片段會建立 Kinesis 來源：

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

- 您需要新增以下 import 陳述式：

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

- 該應用程式使用 Apache Flink S3 接收器寫入 Amazon S3。

  接收器會在輪轉視窗中讀取訊息、將訊息編碼到 S3 儲存貯體物件，並將編碼的物件傳送至 S3 接收器。下列程式碼會對傳送至 Amazon S3 的物件進行編碼：

```
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**
>
> 該應用程式使用 Flink `StreamingFileSink` 物件寫入 Amazon S3。如需有關的詳細資訊`StreamingFileSink`，請參閱《Apache Flink》文件StreamingFileSink中的〈〉。

修改應用程式的程式碼

在本節中，您要修改應用程式的程式碼，以將輸出寫入 Amazon S3 儲存貯體。

使用您的使用者名稱更新下列行，以指定應用程式的輸出位置：

```
private static final String s3SinkPath = "s3a://ka-app-code-<username>/data";
```

編譯應用程式的程式碼

若要編譯應用程式，請執行下列動作：

1. 如果尚未安裝 Java 和 Maven，請安裝。如需詳細資訊，請參閱開始使DataStream 用 (API)教學課程中的必要條件。
2. 使用下列命令編譯應用程式：

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

編譯應用程式會建立應用程式 JAR 檔案 (`target/aws-kinesis-analytics-java-apps-1.0.jar`)。

> **Note**
>
> 提供的來源程式碼依賴於 Java 11 中的程式庫。

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在建立相依資源一節建立的 Amazon S3 儲存貯體。

1. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，導覽至程式碼資料夾，然後選擇「上傳」。
2. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `aws-kinesis-analytics-java-apps-1.0.jar` 檔案。
3. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **`MyApplication`**。
   - 針對執行時間，選擇 Apache Flink。
   - 將版本下拉式清單保留為 Apache Flink 1.15.2 版 (建議版本)。
4. 對於存取許可，選擇建立/更新 IAM 角色 **`kinesis-analytics-MyApplication-us-west-2`**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 在應用程式名稱中，輸入 **`MyApplication`**。
> - 針對執行時間，選擇 Apache Flink。

- 將版本保留為 Apache Flink 版本 1.15.2（建議版本）。

6. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
7. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：`kinesis-analytics-service-MyApplication-us-west-2`
> - 角色：`kinesisanalytics-MyApplication-us-west-2`

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。以您的使用者名稱取代 <username>。

```
{
            "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"
        },

    ]
}
```

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **code/aws-kinesis-analytics-java-apps-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在監控下，確保監控指標層級設為應用程式。
5. 對於CloudWatch 記錄，請選取啟用核取方塊。
6. 選擇更新。

> **Note**
>
> 當您選擇啟用 CloudWatch 記錄時，Apache Flink 的受管理服務會為您建立記錄群組和記錄資料流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`
>
> 此日誌串流用於監控應用程式。這與應用程式用來傳送結果的日誌串流不同。

執行應用程式

1. 在MyApplication頁面上，選擇 [執行]。保持選取不使用快照執行選項，然後確認動作。
2. 應用程式執行時，重新整理頁面。主控台會顯示 Application graph (應用程式圖形)。

驗證應用程式輸出

在 Amazon S3 主控台中開啟 S3 儲存貯體中的 data 資料夾。

幾分鐘後，將顯示包含來自應用程式之彙總資料的物件。

> **Note**
>
> 彙總在 Flink 中預設為啟用。可使用下列命令將其停用：
>
> ```
> sink.producer.aggregation-enabled' = 'false'
> ```

選用：自訂來源和接收器

在本節中，您將自訂來源和接收器物件的設定。

> **Note**
>
> 在變更以下各節中描述的程式碼區段之後，請執行下列動作以重新載入應用程式的程式碼：
>
> - 重複the section called “編譯應用程式的程式碼”一節中的步驟，以編譯更新的應用程式程式碼。
> - 重複the section called “上傳 Apache Flink 串流 Java 程式碼”一節中的步驟，以上傳更新的應用程式程式碼。
> - 在主控台的應用程式頁面上，選擇設定，然後選擇更新，以將更新的應用程式程式碼重新載入您的應用程式。

本區段包含下列各項：

- 設定資料分割
- 設定讀取頻率
- 設定寫入緩衝

設定資料分割

在本節中，您可以設定串流檔案接收器在 S3 儲存貯體中建立的資料夾名稱。若要執行此作業，請將儲存貯體指派者新增至串流檔案接收器。

若要自訂 S3 儲存貯體中建立的資料夾之名稱，請執行以下動作：

1. 將以下 import 陳述式新增到 S3StreamingSinkJob.java 檔案的開頭：

```
import
 org.apache.flink.streaming.api.functions.sink.filesystem.rollingpolicies.DefaultRollingPol
import
 org.apache.flink.streaming.api.functions.sink.filesystem.bucketassigners.DateTimeBucketAss
```

2. 將程式碼中的 createS3SinkFromStaticConfig() 方法更新為如下所示：

```
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().build())
        .build();
    return sink;
}
```

上述程式碼範例使用自訂日期格式的 DateTimeBucketAssigner 在 S3 儲存貯體中建立資料夾。DateTimeBucketAssigner 使用目前的系統時間建立儲存貯體名稱。如果您想要建立自訂值區指派者以進一步自訂建立的資料夾名稱，您可以建立實作BucketAssigner的類別。您可以使用 getBucketId 方法實作自訂邏輯。

BucketAssigner 的自訂實作可以使用 Context 參數取得記錄的詳細資訊，以判定其目的地資料夾。

設定讀取頻率

在本節中，您可以設定來源串流的讀取頻率。

依預設，Kinesis 串流取用者每秒會從來源串流讀取五次。如果有多個用戶端從串流讀取，或應用程式需要重試讀取記錄，則此頻率會造成問題。您可以設定取用者的讀取頻率來避免這些問題。

若要設定 Kinesis 取用者的讀取頻率，請設定 SHARD_GETRECORDS_INTERVAL_MILLIS 設定。

下列程式碼範例會將 SHARD_GETRECORDS_INTERVAL_MILLIS 設定設為 1 秒：

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

設定寫入緩衝

在本節中，您要設定接收器的寫入頻率和其他設定。

依預設，應用程式會每分鐘寫入目的地儲存貯體。您可以設定 DefaultRollingPolicy 物件來變更此間隔和其他設定。

> **Note**
>
> 每次應用程式建立檢查點時，Apache Flink 串流檔案接收器都會寫入其輸出儲存貯體。應用程式預設每分鐘建立一個檢查點。若要增加 S3 接收器的寫入間隔，必須也增加檢查點間隔。

若要設定 DefaultRollingPolicy 物件，請執行下列動作：

1. 增加應用程式的 CheckpointInterval 設定。下列 UpdateApplication動作輸入會將檢查點間隔設定為 10 分鐘：

```
{
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "ConfigurationTypeUpdate" : "CUSTOM",
            "CheckpointIntervalUpdate": 600000
         }
      }
   },
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5
}
```

   若要使用上述程式碼，請指定目前的應用程式版本。您可以使用ListApplications動作擷取應用程式版本。

2. 將以下 import 陳述式新增到 S3StreamingSinkJob.java 檔案的開頭：

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

3. 將 S3StreamingSinkJob.java 檔案中的 createS3SinkFromStaticConfig 方法更新為如下所示：

```
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;
    }
```

上述程式碼範例會將寫入 Amazon S3 儲存貯體的頻率設定為 8 分鐘。

如需關於設定 Apache Flink 串流檔案接收器的詳細資訊，請參閱《Apache Flink 文件》中的資料列編碼格式https://nightlies.apache.org/flink/flink-docs-release-1.13/。

清除 AWS 資源

本節包括清除在 Amazon S3 教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis 資料串流
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台

2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis 資料串流

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面上，選擇刪除 Kinesis 串流，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇 政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

教學課程：使用 Managed Service for Apache Flink 應用程式將 MSK 叢集中某個主題的資料複製到 VPC 中的另一個主題

以下教學課程將示範如何建立 Amazon MSK 叢集和 Amazon VPC 以及兩個主題，以及如何建立一個 Managed Service for Apache Flink 應用程式以從 Amazon MSK 主題讀取並寫入另一個主題。

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成 開始使DataStream 用 (API) 練習。

本教學課程包含下列章節：

- 建立 Amazon VPC 和 Amazon MSK 叢集
- 建立應用程式的程式碼
- 上傳 Apache Flink 串流 Java 程式碼
- 建立應用程式
- 設定應用程式
- 執行應用程式
- 測試應用程式

建立 Amazon VPC 和 Amazon MSK 叢集

若要建立範例 VPC 和 Amazon MSK 叢集以從 Managed Service for Apache Flink 應用程式存取，請按照《Amazon MSK 使用入門》https://docs.aws.amazon.com/msk/latest/developerguide/getting-started.html教學課程進行操作。

完成教學課程時，請注意以下事項：

- 在步驟 3：建立主題中，重複 `kafka-topics.sh --create` 命令以建立名為 `AWSKafkaTutorialTopicDestination` 的目的地主題：

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

- 記錄叢集的啟動伺服器清單。您可以使用下列命令取得啟動程序伺服器清單 (*ClusterArn*以 MSK 叢集的 ARN 取代)：

```
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"
}
```

- 按照教學課程中的步驟進行操作時，請務必在程式碼、命令和主控台項目中使用您選取的 AWS 區域。

建立應用程式的程式碼

在本節中，您會下載並編譯應用程式 JAR 檔案。我們建議使用 Java 11。

此範例的 Java 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 應用程式的程式碼位於 amazon-kinesis-data-analytics-java-examples/KafkaConnectors/KafkaGettingStartedJob.java 檔案中。您可以檢查程式碼，以熟悉 Managed Service for Apache Flink 應用程式程式碼的結構。
4. 使用命令列 Maven 工具或偏好的開發環境來建立 JAR 檔案。若要使用命令列 Maven 工具編譯 JAR 檔案，請輸入下列命令：

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

   如果建置成功，會建立下列檔案：

   ```
   target/KafkaGettingStartedJob-1.0.jar
   ```

   > **Note**
   >
   > 提供的來源程式碼依賴於 Java 11 中的程式庫。如果您使用的是開發環境，

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在開始使DataStream 用 (API)一節建立的 Amazon S3 儲存貯體。

> **Note**
>
> 如果您從入門教學課程中刪除了 Amazon S3 儲存貯體，請再次執行下列 the section called "上傳 Apache Flink 串流 Java 程式碼" 步驟。

1. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
2. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `KafkaGettingStartedJob-1.0.jar` 檔案。
3. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **`MyApplication`**。
   - 針對執行時間，選擇 Apache Flink 1.15.2 版。
4. 對於存取許可，選擇建立/更新 IAM 角色 **`kinesis-analytics-MyApplication-us-west-2`**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：

- 政策：kinesis-analytics-service-*MyApplication*-*us-west-2*
- 角色：kinesisanalytics-*MyApplication*-*us-west-2*

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **KafkaGettingStartedJob-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。

> **Note**
>
> 使用主控台 (例如 CloudWatch 日誌或 Amazon VPC) 指定應用程式資源時，主控台會修改應用程式執行角色，以授與存取這些資源的權限。

4. 在屬性下，選擇新增群組。輸入下列屬性：

群組 ID	金鑰	值
KafkaSource	主題	AWSKafkaTutorialTopic
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**
>
> 預設憑證的 ssl.truststore.password 是「changeit」；如果您使用預設憑證，不需要變更此值。

再次選擇新增群組。輸入下列屬性：

群組 ID	金鑰	值
KafkaSink	主題	AWSKafkaTutorialTopicDestination
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	transaction.timeout.ms	1000

應用程式的程式碼會讀取上述應用程式屬性，以設定用來與 VPC 和 Amazon MSK 叢集互動的來源和接收器。如需關於這些屬性的詳細資訊，請參閱執行時間屬性。

5. 在快照下選擇停用。這可以讓您更輕鬆地更新應用程式，而無需加載無效的應用程式狀態資料。
6. 在監控下，確保監控指標層級設為應用程式。
7. 對於CloudWatch 記錄，請選擇啟用核取方塊。
8. 在虛擬私有雲端 (VPC) 區段中，選擇要與應用程式建立關聯的 VPC。選擇希望應用程式用來存取 VPC 資源的與 VPC 相關聯的子網路和安全群組。
9. 選擇更新。

> **Note**
>
> 當您選擇啟用 CloudWatch 記錄時，Apache Flink 的受管理服務會為您建立記錄群組和記錄資料流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`
>
> 此日誌串流用於監控應用程式。

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

測試應用程式

在本節中，您將記錄寫入來源主題。應用程式會從來源主題讀取記錄，並將其寫入目的地主題。您可以將記錄寫入來源主題並讀取目的地主題中的記錄，以確認應用程式是否正常運作。

若要寫入和讀取主題中的記錄，請按照《Amazon MSK 使用入門》教學課程中的步驟 6：產生和使用資料中的步驟進行操作https://docs.aws.amazon.com/msk/latest/developerguide/getting-started.html。

若要讀取目的地主題，請在與叢集的第二個連線中使用目的地主題而非來源主題的名稱：

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --
consumer.config client.properties --topic AWSKafkaTutorialTopicDestination --from-
beginning
```

如果目的地主題中未顯示任何記錄，請參閱 疑難排解 主題中的無法存取 VPC 中的資源一節。

範例：在 Kinesis 資料串流中使用 EFO 取用者

在本練習中，您會建立 Managed Service for Apache Flink應用程式，該應用程式使用增強型扇出 (EFO) 取用者從 Kinesis 資料串流讀取。如果 Kinesis 取用者使用 EFO，Kinesis Data Streams 服務會提供專屬頻寬，而不是讓取用者與其他從串流讀取的取用者共用串流的固定頻寬。

如需將 EFO 用於 Kinesis 取用者的詳細資訊，請參閱 FLIP-128：將增強型扇出用於 Kinesis 取用者。

您在此範例中建立的應用程式使用 AWS Kinesis 連接器 (flink-connector-kinesis) 1.15.3。

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成 開始使DataStream 用 (API) 練習。

本主題包含下列章節：

- 建立相依資源
- 寫入範例記錄至輸入串流
- 下載並檢查應用程式的程式碼
- 編譯應用程式的程式碼
- 上傳 Apache Flink 串流 Java 程式碼
- 建立並執行 Managed Service for Apache Flink 應用程式
- 清除 AWS 資源

建立相依資源

在為本練習建立 Managed Service for Apache Flink 應用程式之前，先建立下列相依資源：

- 兩個 Kinesis 資料串流 (`ExampleInputStream` 和 `ExampleOutputStream`)
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (`ka-app-code-<username>`)

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需關於建立這些資源的指示，請參閱以下主題：

- 《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。為資料串流 **`ExampleInputStream`** 和 **`ExampleOutputStream`** 命名。
- 《Amazon Simple Storage Service 使用者指南》中的如何建立 S3 儲存貯體？。透過附加登入名稱 (例如 **`ka-app-code-<username>`**)，為 Amazon S3 儲存貯體提供全域唯一的名稱。

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

1. 使用下列內容建立名為 stock.py 的檔案：

```
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. 執行 stock.py 指令碼：

```
$ python stock.py
```

在完成教學課程的其餘部分時，讓指令碼保持執行狀態。

下載並檢查應用程式的程式碼

此範例的 Java 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 請前往 `amazon-kinesis-data-analytics-java-examples/EfoConsumer` 目錄。

應用程式的程式碼位於 `EfoApplication.java` 檔案中。請留意下列與應用程式的程式碼相關的資訊：

- 您可以在 Kinesis 取用者上設定下列參數來啟用 EFO 取用者：
  - RECORD_PUBLISHER_TYPE：將此參數設定為 EFO，以便讓應用程式使用 EFO 取用者來存取 Kinesis 資料串流資料。
  - EFO_CONSUMER_NAME：將此參數設定為字串值，確保在此串流的取用者中保持唯一。在相同的 Kinesis 資料串流中重複使用取用者名稱，將導致先前使用該名稱的使用者遭到終止。
- 下列程式碼範例示範如何將值指派給取用者組態屬性，以便使用 EFO 取用者從來源串流讀取：

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

編譯應用程式的程式碼

若要編譯應用程式，請執行下列動作：

1. 如果尚未安裝 Java 和 Maven，請安裝。如需詳細資訊，請參閱開始使DataStream 用 (API)教學課程中的必要條件。
2. 使用下列命令編譯應用程式：

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

> **Note**
>
> 提供的來源程式碼依賴於 Java 11 中的程式庫。

編譯應用程式會建立應用程式 JAR 檔案 (`target/aws-kinesis-analytics-java-apps-1.0.jar`)。

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在建立相依資源一節建立的 Amazon S3 儲存貯體。

1. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
2. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `aws-kinesis-analytics-java-apps-1.0.jar` 檔案。
3. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **`MyApplication`**。
   - 針對執行時間，選擇 Apache Flink。

     > **Note**
     >
     > Managed Service for Apache Flink 使用 Apache Flink 1.15.2 版。

   - 將版本下拉式清單保留為 Apache Flink 1.15.2 版 (建議版本)。

4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：kinesis-analytics-service-*MyApplication*-*us-west-2*
> - 角色：kinesisanalytics-*MyApplication*-*us-west-2*

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

> **Note**
>
> 這些許可授與應用程式存取 EFO 取用者的能力。

```
{
    "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:*"
            ]
        }
    ]
}
```

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **aws-kinesis-analytics-java-apps-1.0.jar**。

3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在屬性下，選擇建立群組。
5. 輸入以下應用程式屬性和數值：

群組 ID	金鑰	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. 在屬性下，選擇建立群組。
7. 輸入以下應用程式屬性和數值：

群組 ID	金鑰	值
ProducerConfigProperties	**OUTPUT_STREAM**	**ExampleOutputStream**
ProducerConfigProperties	**AWS_REGION**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

8. 在監控下，確保監控指標層級設為應用程式。
9. 對於CloudWatch 記錄，請選取啟用核取方塊。

10. 選擇更新。

> **Note**
>
> 當您選擇啟用 CloudWatch 記錄時，Apache Flink 的受管理服務會為您建立記錄群組和記錄資料流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`
>
> 此日誌串流用於監控應用程式。這與應用程式用來傳送結果的日誌串流不同。

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

您可以在 CloudWatch 主控台上檢查 Apache Flink 的受管理服務量度，以確認應用程式是否正常運作。

您也可以在資料串流的增強型散發索引標籤中，查看 Kinesis 資料串流主控台，以取得用戶名稱 () basic-efo-flink-app。

清除 AWS 資源

本節包括清除在 efo 視窗教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis 資料串流
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台

2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis 資料串流

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇 ExampleOutputStream，選擇動作，選擇刪除，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。

4. 選擇刪除日誌群組，然後確認刪除。

範例：寫入 Kines is Data Firehose

在本練習中，您會建立一個 Managed Service for Apache Flink 應用程式，該應用程式將 Kinesis 資料串流作為來源，將 Kinesis Data Firehose 串流作為接收器。使用接收器，您就可以驗證 Amazon S3 儲存貯體的應用程式輸出。

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成 開始使DataStream 用 (API) 練習。

本節包含下列步驟：

- 建立相依資源
- 寫入範例記錄至輸入串流
- 下載並檢查 Apache Flink 串流 Java 程式碼
- 編譯應用程式的程式碼
- 上傳 Apache Flink 串流 Java 程式碼
- 建立並執行 Managed Service for Apache Flink 應用程式
- 清除 AWS 資源

建立相依資源

為本練習建立 Managed Service of Apache Flink 之前，請先建立下列相依資源：

- Kinesis 資料串流 (`ExampleInputStream`)
- 應用程式將輸出寫入 (`ExampleDeliveryStream`) 的 Kinesis Data Firehose 串流。
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (`ka-app-code-<username>`)

您可以在主控台中建立 Kinesis 串流、Amazon S3 儲存貯體和 Kinesis Data Firehose 串流。如需關於建立這些資源的指示，請參閱以下主題：

- 《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。為資料串流 **`ExampleInputStream`** 命名。

- 《Amazon Kinesis Data Firehose 開發人員指南》中的建立 Amazon Kinesis Data Firehose 交付串流。為 Kinesis Data Firehose **ExampleDeliveryStream** 串流命名。當您建立 Kinesis Data Firehose 串流時，也會建立 Kinesis Data Firehose 串流的 S3 目的地和 IAM 角色。
- 《Amazon Simple Storage Service 使用者指南》中的如何建立 S3 儲存貯體？。透過附加登入名稱 (例如 **ka-app-code-*<username>***)，為 Amazon S3 儲存貯體提供全域唯一的名稱。

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

1. 使用下列內容建立名為 `stock.py` 的檔案：

```
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. 執行 stock.py 指令碼：

```
$ python stock.py
```

在完成教學課程的其餘部分時，讓指令碼保持執行狀態。

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的 Java 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 使用以下指令複製遠端儲存庫：

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

2. 請前往 amazon-kinesis-data-analytics-java-examples/FirehoseSink 目錄。

應用程式的程式碼位於 FirehoseSinkStreamingJob.java 檔案中。請留意下列與應用程式的程式碼相關的資訊：

- 應用程式使用 Kinesis 來源從來源串流讀取。以下程式碼片段會建立 Kinesis 來源：

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

- 應用程式使用 Kinesis Data Firehose 接收器將資料寫入 Kinesis Data Firehose 串流。以下程式碼片段會建立 Kinesis Data Firehose 接收器：

```
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();
```

```
    }
```

編譯應用程式的程式碼

若要編譯應用程式，請執行下列動作：

1. 如果尚未安裝 Java 和 Maven，請安裝。如需詳細資訊，請參閱開始使DataStream 用 (API)教學課程中的必要條件。
2. 若要將 Kinesis 連接器用於以下應用程式，您需要下載、建置並安裝 Apache Maven。如需詳細資訊，請參閱the section called “將 Apache Flink Kinesis 串流連接器與 Apache Flink 先前版本搭配使用”。
3. 使用下列命令編譯應用程式：

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

> **Note**
>
> 提供的來源程式碼依賴於 Java 11 中的程式庫。

編譯應用程式會建立應用程式 JAR 檔案 (`target/aws-kinesis-analytics-java-apps-1.0.jar`)。

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在建立相依資源一節建立的 Amazon S3 儲存貯體。

上傳應用程式的程式碼

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在主控台中，選擇 [ka-app-code- <username>值區]，然後選擇 [上傳]。
3. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `java-getting-started-1.0.jar` 檔案。
4. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

您可以使用主控台或 AWS CLI 建立和執行 Managed Service for Apache Flink 應用程式。

> **Note**
>
> 使用主控台建立應用程式時，系統會為您建立 AWS Identity and Access Management (IAM) 和 Amazon CloudWatch 日誌資源。當您使用 AWS CLI 建立應用程式時，則會個別建立這些資源。

主題

建立和執行應用程式 (主控台)

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 對於 Description (說明)，輸入 **My java test app**。
   - 針對執行時間，選擇 Apache Flink。

     > **Note**
     >
     > Managed Service for Apache Flink 使用 Apache Flink 1.15.2 版。

   - 將版本下拉式清單保留為 Apache Flink 1.15.2 版 (建議版本)。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式會使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - 角色：`kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流和 Kinesis Data Firehose 串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*) 的所有執行個體。

```
{
    "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"
        }
    ]
}
```

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **java-getting-started-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在監控下，確保監控指標層級設為應用程式。
5. 對於CloudWatch 記錄，請選取啟用核取方塊。
6. 選擇更新。

> **Note**
>
> 當您選擇啟用 CloudWatch 記錄時，Apache Flink 的受管理服務會為您建立記錄群組和記錄資料流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

停止應用程式

在MyApplication頁面上，選擇 [停止]。確認動作。

更新應用程式

您可以使用主控台更新應用程式設定，例如應用程式屬性、監控設定及位置或應用程式 JAR 的檔名。

在MyApplication頁面上，選擇設定。更新應用程式設定，然後選擇更新。

> **Note**
>
> 若要在主控台上更新應用程式的程式碼，您必須變更 JAR 的物件名稱、使用不同的 S3 儲存貯體，或使用the section called “更新應用程式的程式碼”章節中所述的 AWS CLI。如果檔案名稱或儲存貯體未變更，當您在設定頁面上選擇更新時，不會重新載入應用程式的程式碼。

建立並執行應用程式 (AWS CLI)

在本節中，您可以使用 AWS CLI 建立和執行 Managed Service for Apache Flink 應用程式。

建立許可政策

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 `read` 動作的許可，而另一條則是授與目的地串流上 `write` 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

使用以下程式碼來建立 `AKReadSourceStreamWriteSinkStream` 許可政策。以您用於建立 Amazon S3 儲存貯體 (以儲存應用程式的程式碼) 的 *`username`* 來取代。以您的帳戶 ID 取代 Amazon Resource Names (ARNs) (*`012345678901`*) 中的帳戶 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"
        }
    ]
}
```

如需建立許可政策的指 step-by-step 示，請參閱 IAM 使用指南中的教學課程：建立和附加您的第一個客戶受管政策。

> **Note**
>
> 若要存取其他 Amazon 服務，您可以使用 AWS SDK for Java。Managed Service for Apache Flink 自動將 SDK 所需的憑證設定為與應用程式相關聯的服務執行 IAM 角色。無須採取額外的步驟。

建立 IAM 角色

在本節中，您會建立 Managed Service for Apache Flink 應用程式可以擔任的 IAM 角色，以便讀取來源串流與寫入目的地串流。

Managed Service for Apache Flink 如果沒有許可，將無法存取串流。您可以透過 IAM 角色來授與這些許可。各 IAM 角色都有連接兩項政策。信任政策會授與 Managed Service for Apache Flink 許可以擔任角色。許可政策決定了 Managed Service for Apache Flink 在擔任角色之後可以執行的作業。

您會將在上一節中建立的許可政策連接至此角色。

若要建立一個 IAM 角色

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Roles (角色) 、Create Role (建立角色)。
3. 在選取可信身分類型下，選擇 AWS 服務。在 Choose the service that will use this role (選擇將使用此角色的服務) 下，選擇 Kinesis。在 Select your use case (選取您的使用案例) 下，選擇 Kinesis Analytics (Kinesis 分析)。

   選擇 Next: Permissions (下一步：許可)。

4. 在 Attach permissions policies (連接許可政策) 頁面上，選擇 Next: Review (下一步：檢閱)。您會在建立角色後連接許可政策。
5. 在建立角色頁面，輸入 **MF-stream-rw-role** 作為角色名稱。選擇建立角色。

   現在您已建立新的 IAM 角色，名為 MF-stream-rw-role。您接著會更新角色的信任和許可政策。
6. 將 許可政策連接到角色。

   > **Note**
   >
   > 在此練習中，Managed Service for Apache Flink 擔任從 Kinesis 資料串流 (來源) 讀取資料並將輸出寫入另一個 Kinesis 資料串流的角色。因此您會連接在上一個步驟中建立的政策，the section called “建立許可政策”。

   a. 在摘要頁面上，選擇許可標籤。
   b. 選擇 Attach Policies (連接政策)。
   c. 在搜尋方塊中，輸入 **AKReadSourceStreamWriteSinkStream** (您在上一節中建立的政策)。
   d. 選擇 AK 原ReadSourceStreamWriteSinkStream則，然後選擇 [附加原則]。

您現在已建立應用程式將用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的指 step-by-step 示，請參閱 IAM 使用者指南中的建立 IAM 角色 (主控台)。

建立 Managed Service for Apache Flink 應用程式

1. 將下列 JSON 程式碼複製到名為 create_request.json 的檔案。以您之前建立之角色的 ARN，取代範例角色 ARN。以您在the section called “建立相依資源”一節中選擇的尾碼 (ka-app-code-*<username>*) 取代儲存貯體 ARN 尾碼。以您的帳戶 ID 取代服務執行角色中的範例帳戶 ID (*012345678901*)。

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-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. 以建立應用程式的上述請求，執行 CreateApplication 動作：

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

現在已建立應用程式。您會在下一個步驟中啟動應用程式。

啟動應用程式

在本節中，您會透過 StartApplication 動作來啟動應用程式。

啟動應用程式

1. 將下列 JSON 程式碼複製到名為 start_request.json 的檔案。

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

2. 以啟動應用程式的上述請求，執行 StartApplication 動作：

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

應用程式現在正在執行。您可以在 Amazon CloudWatch 主控台上查看 Apache Flink 的受管服務指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 StopApplication 動作來停止應用程式。

停止應用程式

1. 將下列 JSON 程式碼複製到名為 stop_request.json 的檔案。

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

2. 以停止應用程式的上述請求，執行 StopApplication 動作：

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

現在已停止應用程式。

新增記 CloudWatch 錄選項

您可以使用AWS CLI將 Amazon CloudWatch 日誌串流新增至您的應用程式。如需搭配應用程式使用 CloudWatch 記錄的相關資訊，請參閱the section called “設定日誌”。

更新應用程式的程式碼

需要使用新版本的程式碼套件更新應用程式的程式碼時，請使用 UpdateApplicationAWS CLI 動作。

若要使用 AWS CLI，請從 Amazon S3 儲存貯體刪除先前的程式碼套件，上傳新版本，然後呼叫 UpdateApplication，指定相同的 Amazon S3 儲存貯體和物件名稱。

UpdateApplication 動作的下列範例請求會重新載入應用程式的程式碼並重新啟動應用程式。將 CurrentApplicationVersionId 更新至目前的應用程式版本。您可以使用 ListApplications 或 DescribeApplication 動作來檢查目前的應用程式版本。使用您在the section called “建立相依資源”一節中選擇的尾碼更新儲存貯體名稱尾碼 (<*username*>)。

```
{
    "ApplicationName": "test",
```

```
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "java-getting-started-1.0.jar"
                }
            }
        }
    }
}
```

清除 AWS 資源

本節包括清除在入門教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis 資料串流
- 刪除 Kinesis Data Firehose 串流
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 選擇設定。
4. 在快照區段中，選擇停用，然後選擇更新。
5. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis 資料串流

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。

3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。

刪除 Kinesis Data Firehose 串流

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在「Kinesis Data Firehose」面板中，選擇。ExampleDeliveryStream
3. 在ExampleDeliveryStream頁面中，選擇「刪除 Kinesis Data Firehose 串流」，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。
4. 如果您為 Kinesis Data Firehose 串流的目的地建立了 Amazon S3 儲存貯體，請一併刪除該儲存貯體。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 如果您為 Kinesis Data Firehose 串流建立了新政策，請一併刪除該政策。
7. 在導覽列中，選擇角色。
8. 選擇運動分析-MyApplication 美西 -2 角色。
9. 選擇刪除角色，然後確認刪除。
10. 如果您為 Kinesis Data Firehose 串流建立了新角色，請一併刪除該角色。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。

3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：從不同帳戶的 Kinesis 串流中讀取

本範例示範如何建立可從不同帳戶的 Kinesis 串流讀取資料的 Managed Service for Apache Flink 應用程式。在此範例中，您將一個帳戶用於來源 Kinesis 串流，將另一個帳戶用於 Managed Service for Apache Flink 應用程式和接收器 Kinesis 串流。

本主題包含下列章節：

- 必要條件
- 設定
- 建立來源 Kinesis 串流
- 建立和更新 IAM 角色和政策
- 更新 Python 指令碼
- 更新 Java 應用程式
- 建置、上傳並執行應用程式

必要條件

- 在本教學課程中，您將修改入門範例，從不同帳戶的 Kinesis 串流讀取資料。請完成開始使 DataStream 用 (API)教學課程後再繼續。
- 您需要兩個 AWS 帳戶才能完成本教學課程：一個用於來源串流，另一個用於應用程式和接收器串流。將用於入門教學課程的 AWS 帳戶用於應用程式和接收串流。對來源串流使用不同的 AWS 帳戶。

設定

您將使用具名設定檔存取兩個 AWS 帳戶。修改您的 AWS 憑證和組態檔案以包含兩個設定檔，這兩個設定檔包含兩個帳戶的地區和連線資訊。

下列範例憑證檔案包含兩個具名的設定檔 `ka-source-stream-account-profile` 和 `ka-sink-stream-account-profile`。將用於入門教學課程的帳戶用於接收器串流帳戶。

```
[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
```

下列範例組態檔案包含具有區域和輸出格式資訊的相同具名設定檔。

```
[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**
>
> 本教學課程不使用 `ka-sink-stream-account-profile`。它作為如何使用設定檔存取兩個不同 AWS 帳戶的範例。

如需關於在 AWS CLI 中使用具名設定檔的詳細資訊，請參閱《AWS Command Line Interface 文件》中的具名設定檔。

建立來源 Kinesis 串流

在本節中，您將在來源帳戶中建立 Kinesis 串流。

輸入下列命令建立 Kinesis 串流，供應用程式用來輸入。請注意，`--profile` 參數會指定要使用的帳戶設定檔。

```
$ aws kinesis create-stream \
--stream-name SourceAccountExampleInputStream \
--shard-count 1 \
--profile ka-source-stream-account-profile
```

建立和更新 IAM 角色和政策

若要允許跨 AWS 帳戶進行物件存取，請在來源帳戶中建立 IAM 角色和政策。然後修改接收器帳戶中的 IAM 政策。如需關於建立和管理 IAM 角色和政策的詳細資訊，請參閱《AWS Identity and Access Management 使用者指南》中的下列主題：

- 建立 IAM 角色
- 建立 IAM 政策

接收器帳戶角色和策略

1. 從入門教學課程編輯 `kinesis-analytics-service-MyApplication-us-west-2` 策略。此政策允許應用程式擔任來源帳戶中的角色，以讀取來源串流。

   > **Note**
   >
   > 使用主控台建立應用程式時，主控台會建立名為 `kinesis-analytics-service-<application name>-<application region>` 的政策和名為 `kinesisanalytics-<application name>-<application region>` 的角色。

   將下面反白顯示的部分新增至政策。以要用於來源串流的帳戶 ID 取代範例帳戶ID (*SOURCE01234567*)。

```
{
    "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. 開啟 kinesis-analytics-MyApplication-us-west-2 角色，記下其 Amazon Resource Name (ARN)。您會在下一節中用到它。角色 ARN 如下所示：

```
arn:aws:iam::SINK012345678:role/service-role/kinesis-analytics-MyApplication-us-
west-2
```

來源帳戶角色和策略

1. 在名為的 KA-Source-Stream-Policy 來源帳戶中建立策略。可以為政策使用下列 JSON。以來源帳戶 ID 取代範例帳戶 ID。

```
{
    "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. 在名為的 MF-Source-Stream-Role 來源帳戶中建立角色。執行下列動作，以使用受管 Flink 使用案例建立角色：

    1. 在 IAM 管理主控台中，選擇建立角色。
    2. 在建立角色頁面上選擇 AWSAWS 服務。在服務清單中，選擇 Kinesis。
    3. 在選取您的使用案例區塊中，選擇 Managed Service for Apache Flink。
    4. 選擇 Next: Permissions (下一步：許可)。
    5. 新增您在上個步驟中建立的 KA-Source-Stream-Policy 許可政策：選擇 Next: Add Tags (下一步：新增標籤)。
    6. 選擇 下一步：檢閱。
    7. 將角色命名為 KA-Source-Stream-Role。您的應用程式將使用此角色來存取來源串流。

3. 將接收器帳戶的 kinesis-analytics-MyApplication-us-west-2 ARN 新增至來源帳戶中 KA-Source-Stream-Role 角色的信任關係中：

   1. 在 IAM 主控台中開啟 KA-Source-Stream-Role。
   2. 選取 Trust Relationships (信任關係) 索引標籤。
   3. 選擇 Edit trust relationship (編輯信任關係)。
   4. 將下列程式碼用於信任關係。以您的接收器帳戶 ID 取代範例帳戶 ID (*SINK012345678*)。

```
{
    "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 指令碼

在本節中，您將更新產生範例資料的 Python 指令碼，以使用來源帳戶設定檔。

使用下列反白顯示的變更更新 stock.py 指令碼。

```
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 應用程式

在本節中，您將更新 Java 應用程式的程式碼，以便在從來源串流讀取時擔任來源帳戶角色。

對 BasicStreamingJob.java 檔案進行下列變更。以您的來源帳戶 ID (*SOURCE01234567*) 取代範例來源帳戶 ID (SOURCE01234567)。

```
package com.amazonaws.services.managed-flink;

import com.amazonaws.services.managed-flink.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 Managed Service for Apache Flink 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");
    }
}
```

建置、上傳並執行應用程式

執行下列動作以更新並執行應用程式：

1. 在包含 pom.xml 檔案的目錄中，執行下列命令來再次建置應用程式。

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

2. 從 Amazon Simple Storage Service (Amazon S3) 儲存貯體刪除先前的 JAR 檔案，然後將新 aws-kinesis-analytics-java-apps-1.0.jar 檔案上傳到 S3 儲存貯體。
3. 在 Managed Service for Apache Flink 主控台的應用程式頁面中，依序選擇設定和更新，以重新載入應用程式 JAR 檔案。
4. 執行stock.py 指令碼，以將資料傳送至來源串流。

   ```
   python stock.py
   ```

應用程式現在會從另一個帳戶的 Kinesis 串流讀取資料。

您可以檢查 ExampleOutputStream 串流的 PutRecords.Bytes 指標，以驗證應用程式是否正在運作。如果輸出串流中有活動，表示應用程式運作正常。

教學課程：搭配 Amazon MSK 使用自訂信任存放區

目前資料來源 API

如果您使用目前的資料來源 API，您的應用程式可以利用此處所述的 MSK Config Providers 公用程式。這可讓您的 KafkaSource 函數在 Amazon S3 中存取相互 TLS 的金鑰儲存庫和信任存放區。

```
...
// define names of config providers:
builder.setProperty("config.providers", "secretsmanager,s3import");
```

```
// provide implementation classes for each provider:
builder.setProperty("config.providers.secretsmanager.class",
 "com.amazonaws.kafka.config.providers.SecretsManagerConfigProvider");
builder.setProperty("config.providers.s3import.class",
 "com.amazonaws.kafka.config.providers.S3ImportConfigProvider");

String region = appProperties.get(Helpers.S3_BUCKET_REGION_KEY).toString();
String keystoreS3Bucket = appProperties.get(Helpers.KEYSTORE_S3_BUCKET_KEY).toString();
String keystoreS3Path = appProperties.get(Helpers.KEYSTORE_S3_PATH_KEY).toString();
String truststoreS3Bucket =
 appProperties.get(Helpers.TRUSTSTORE_S3_BUCKET_KEY).toString();
String truststoreS3Path = appProperties.get(Helpers.TRUSTSTORE_S3_PATH_KEY).toString();
String keystorePassSecret =
 appProperties.get(Helpers.KEYSTORE_PASS_SECRET_KEY).toString();
String keystorePassSecretField =
 appProperties.get(Helpers.KEYSTORE_PASS_SECRET_FIELD_KEY).toString();

// region, etc..
builder.setProperty("config.providers.s3import.param.region", region);

// properties
builder.setProperty("ssl.truststore.location", "${s3import:" + region + ":" +
 truststoreS3Bucket + "/" + truststoreS3Path + "}");
builder.setProperty("ssl.keystore.type", "PKCS12");
builder.setProperty("ssl.keystore.location", "${s3import:" + region + ":" +
 keystoreS3Bucket + "/" + keystoreS3Path + "}");
builder.setProperty("ssl.keystore.password", "${secretsmanager:" + keystorePassSecret +
 ":" + keystorePassSecretField + "}");
builder.setProperty("ssl.key.password", "${secretsmanager:" + keystorePassSecret + ":"
 + keystorePassSecretField + "}");
...
```

詳細資訊和逐步導覽可以在此處找到。

舊版 SourceFunction API

如果您使用舊版 SourceFunction API，您的應用程式將使用自訂序列化和還原序列化結構描述，這些結構描述會覆寫載入自訂信任存放區的open方法。這讓應用程式在重新啟動或取代執行緒之後可以使用信任存放區。

可使用以下程式碼檢索和存儲自訂信任存放區：

```
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 要求信任存放區為 JKS 格式。

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成 開始使DataStream 用 (API) 練習。

以下教學課程示範如何安全地連線 (傳輸中加密) 到使用由自訂、私有甚至自託管憑證授權機構 (CA) 發行的伺服器憑證之 Kafka 叢集。

為了透過 TLS 將任何 Kafka 用戶端安全地連線到 Kafka 叢集，Kafka 用戶端 (如範例 Flink 應用程式) 必須信任由 Kafka 叢集的伺服器憑證 (從發行 CA 到根層級 CA) 提供的完整信任鏈。作為自訂信任存放區的範例，我們將使用啟用雙向 TLS (MTLS) 身分驗證的 Amazon MSK 叢集。這表示 MSK 叢集節點會使用由 AWS Certificate Manager 私有憑證授權機構 (ACM Private CA) 發行的伺服器憑證，該憑證對您的帳戶和區域是私有的，因此不受執行 Flink 應用程式之 Java 虛擬機器 (JVM) 的預設信任存放區所信任。

> **Note**
>
> - 金鑰存放區用於存放應用程式應該提供給伺服器或用戶端以進行驗證的私有金鑰和身分憑證。
> - 信任存放區用於存放來自憑證授權機構 (CA) 的憑證，以驗證伺服器在 SSL 連線中提供的憑證。

您也可以使用本教學課程中的技術，在 Managed Service for Apache Flink 應用程式與其他 Apache Kafka 來源之間進行互動，例如：

- 託管在AWS (Amazon EC2 或 Amazon EKS) 中的自訂 Apache Kafka 叢集
- 託管在 AWS 中的 Confluent Kafka 叢集
- 透過 AWS Direct Connect 或 VPN 存取的內部部署 Kafka 叢集

本教學課程包含下列章節：

- 建立 VPC 和 Amazon MSK 叢集
- 建立自訂信任存放區並將其套用至叢集
- 建立應用程式的程式碼
- 上傳 Apache Flink 串流 Java 程式碼
- 建立應用程式
- 設定應用程式
- 執行應用程式
- 測試應用程式

建立 VPC 和 Amazon MSK 叢集

若要建立範例 VPC 和 Amazon MSK 叢集以從 Managed Service for Apache Flink 應用程式存取，請按照《Amazon MSK 使用入門》https://docs.aws.amazon.com/msk/latest/developerguide/getting-started.html教學課程進行操作。

完成教學課程時，請注意以下事項：

- 在步驟 3：建立主題中，重複 `kafka-topics.sh --create` 命令以建立名為 `AWSKafkaTutorialTopicDestination` 的目的地主題：

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

> **Note**
>
> 如果kafka-topics.sh 命令傳回 ZooKeeperClientTimeoutException，請確認 Kafka 叢集的安全群組具有一項傳入規則，即允許來自用戶端執行個體私有 IP 位址的所有流量。

- 記錄叢集的啟動伺服器清單。您可以使用下列命令取得啟動程序伺服器清單 (*ClusterArn*以 MSK 叢集的 ARN 取代)：

```
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"
}
```

- 按照本教學課程和先決教學課程中的步驟進行操作時，請務必在程式碼、命令和主控台項目中使用您選取的 AWS 區域。

建立自訂信任存放區並將其套用至叢集

在本節中，您將建立自訂憑證授權機構 (CA)、使用它來產生自訂信任存放區，並將其套用至 MSK 叢集。

若要建立和套用您的自訂信任存放區，請遵循 《Amazon Managed Streaming for Apache Kafka 開發人員指南》中的用戶端身分驗證教學課程。

建立應用程式的程式碼

在本節中，您會下載並編譯應用程式 JAR 檔案。

此範例的 Java 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 應用程式的程式碼位於 amazon-kinesis-data-analytics-java-examples/CustomKeystore 檔案中。您可以檢查程式碼，以熟悉 Managed Service for Apache Flink 程式碼的結構。
4. 使用命令列 Maven 工具或偏好的開發環境來建立 JAR 檔案。若要使用命令列 Maven 工具編譯 JAR 檔案，請輸入下列命令：

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

如果建置成功，會建立下列檔案：

```
target/flink-app-1.0-SNAPSHOT.jar
```

> **Note**
>
> 提供的來源程式碼依賴於 Java 11 中的程式庫。

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會將應用程式的程式碼上傳至在開始使DataStream 用 (API)一節建立的 Amazon S3 儲存貯體。

> **Note**
>
> 如果您從入門教學課程中刪除了 Amazon S3 儲存貯體，請再次執行下列 the section called “上傳 Apache Flink 串流 Java 程式碼” 步驟。

1. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
2. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 flink-app-1.0-SNAPSHOT.jar 檔案。
3. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 針對執行時間，選擇 Apache Flink 1.15.2 版。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - 角色：`kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **flink-app-1.0-SNAPSHOT.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。

> **Note**
>
> 當您使用主控台 (例如日誌或 VPC) 指定應用程式資源時，主控台會修改您的應用程式執行角色，以授與存取這些資源的許可。

4. 在屬性下，選擇新增群組。輸入下列屬性：

群組 ID	金鑰	值
KafkaSource	主題	AWSKafkaTutorialTopic
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**
>
> 預設憑證的 ssl.truststore.password 是「changeit」；如果您使用預設憑證，不需要變更此值。

再次選擇新增群組。輸入下列屬性：

群組 ID	金鑰	值
KafkaSink	主題	AWSKafkaTutorialTopicDestination
KafkaSink	bootstrap.servers	###########
KafkaSink	security.protocol	SSL

群組 ID	金鑰	值
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSink	ssl.truststore.password	changeit
KafkaSink	transaction.timeout.ms	1000

應用程式的程式碼會讀取上述應用程式屬性，以設定用來與 VPC 和 Amazon MSK 叢集互動的來源和接收器。如需關於這些屬性的詳細資訊，請參閱執行時間屬性。

5. 在快照下選擇停用。這可以讓您更輕鬆地更新應用程式，而無需加載無效的應用程式狀態資料。
6. 在監控下，確保監控指標層級設為應用程式。
7. 對於CloudWatch 記錄，請選擇啟用核取方塊。
8. 在虛擬私有雲端 (VPC) 區段中，選擇要與應用程式建立關聯的 VPC。選擇希望應用程式用來存取 VPC 資源的與 VPC 相關聯的子網路和安全群組。
9. 選擇更新。

> **Note**
>
> 當您選擇啟用 CloudWatch 記錄時，Apache Flink 的受管理服務會為您建立記錄群組和記錄資料流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`
>
> 此日誌串流用於監控應用程式。

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

測試應用程式

在本節中，您將記錄寫入來源主題。應用程式會從來源主題讀取記錄，並將其寫入目的地主題。您可以將記錄寫入來源主題並讀取目的地主題中的記錄，以確認應用程式是否正常運作。

若要寫入和讀取主題中的記錄，請按照《Amazon MSK 使用入門》教學課程中的步驟 6：產生和使用資料中的步驟進行操作https://docs.aws.amazon.com/msk/latest/developerguide/getting-started.html。

若要讀取目的地主題，請在與叢集的第二個連線中使用目的地主題而非來源主題的名稱：

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --
consumer.config client.properties --topic AWSKafkaTutorialTopicDestination --from-
beginning
```

如果目的地主題中未顯示任何記錄，請參閱 疑難排解 主題中的無法存取 VPC 中的資源一節。

Python 範例

下列範例示範如何搭配使用 Python 與 Apache Flink 資料表 API 建立應用程式。

主題

- 範例：在 Python 中建立輪轉視窗
- 範例：在 Python 中建立滑動視窗
- 範例：使用 Python 將串流資料傳送至 Amazon S3

範例：在 Python 中建立輪轉視窗

在本練習中，您將使用輪轉視窗建立可彙總資料的適用於 Python 的 Managed Service for Apache Flink 應用程式。

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成 入門 (Python) 練習。

本主題包含下列章節：

- 建立相依資源
- 寫入範例記錄至輸入串流

- 下載並檢查應用程式的程式碼
- 壓縮並上傳 Apache Flink 串流 Python 程式碼
- 建立並執行 Managed Service for Apache Flink 應用程式
- 清除 AWS 資源

建立相依資源

在為本練習建立 Managed Service for Apache Flink 應用程式之前，先建立下列相依資源：

- 兩個 Kinesis 資料串流 (`ExampleInputStream` 和 `ExampleOutputStream`)
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (`ka-app-code-<username>`)

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需關於建立這些資源的指示，請參閱以下主題：

- 《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。為資料串流 **`ExampleInputStream`** 和 **`ExampleOutputStream`** 命名。
- 《Amazon Simple Storage Service 使用者指南》中的如何建立 S3 儲存貯體？。透過附加登入名稱 (例如 **`ka-app-code-<username>`**)，為 Amazon S3 儲存貯體提供全域唯一的名稱。

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

> **Note**
>
> 本節中的 Python 指令碼會使用 AWS CLI。您必須將 AWS CLI 設定為使用您的帳戶憑證和預設區域。若要設定 AWS CLI，請輸入下列內容：
>
> ```
> aws configure
> ```

1. 使用下列內容建立名為 stock.py 的檔案：

```
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. 執行 stock.py 指令碼：

```
$ python stock.py
```

在完成教學課程的其餘部分時，讓指令碼保持執行狀態。

下載並檢查應用程式的程式碼

此範例的 Python 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 請前往 amazon-kinesis-data-analytics-java-examples/python/TumblingWindow 目錄。

應用程式的程式碼位於 tumbling-windows.py 檔案中。請留意下列與應用程式的程式碼相關的資訊：

- 應用程式使用 Kinesis 資料表來源從來源串流讀取。下列程式碼片段會呼叫 create_table 函數來建立 Kinesis 資料表來源：

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

create_table 函數使用 SQL 命令來建立由串流來源支援的資料表：

```
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)
```

- 應用程式使用 Tumble 運算子來彙總指定的輪轉視窗中的記錄，並將彙總記錄作為資料表物件傳回：

```
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")
```

- 應用程式使用 Kinesis Flink 連接器，來自 flink-sql-connector-kinesis-1.15.2.jar

壓縮並上傳 Apache Flink 串流 Python 程式碼

在本節中，您會將應用程式的程式碼上傳至在建立相依資源一節建立的 Amazon S3 儲存貯體。

1. 使用偏好的壓縮應用程式來壓縮 tumbling-windows.py 和 flink-sql-connector-kinesis-1.15.2.jar 檔案。命名存檔 myapp.zip。
2. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
3. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 myapp.zip 檔案。
4. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 針對執行時間，選擇 Apache Flink。

> **Note**
>
> Managed Service for Apache Flink 使用 Apache Flink 1.15.2 版。

- 將版本下拉式清單保留為 Apache Flink 1.15.2 版 (建議版本)。

4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：kinesis-analytics-service-*MyApplication*-*us-west-2*
> - 角色：kinesisanalytics-*MyApplication*-*us-west-2*

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **myapp.zip**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在屬性下，選擇新增群組。
5. 輸入下列資料：

群組 ID	金鑰	值
consumer.config.0	**input.stream.name**	**ExampleInputStream**

群組 ID	金鑰	值
consumer.config.0	aws.region	us-west-2
consumer.config.0	scan.stream.initpos	LATEST

選擇儲存。

6. 在屬性下，再次選擇新增群組。
7. 輸入下列資料：

群組 ID	金鑰	值
producer.config.0	output.stream.name	ExampleOutputStream
producer.config.0	aws.region	us-west-2
producer.config.0	shard.count	1

8. 在屬性下，再次選擇新增群組。針對群組 ID，輸入 **kinesis.analytics.flink.run.options**。這個特殊的屬性群組會告訴您的應用程式在何處尋找其程式碼資源。如需詳細資訊，請參閱 指定程式碼檔案。
9. 輸入下列資料：

群組 ID	金鑰	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. 在監控下，確保監控指標層級設為應用程式。
11. 對於CloudWatch 記錄，請選取啟用核取方塊。
12. 選擇更新。

> **Note**
>
> 當您選擇啟用 CloudWatch 記錄時，Apache Flink 的受管理服務會為您建立記錄群組和記錄資料流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`
>
> 此日誌串流用於監控應用程式。這與應用程式用來傳送結果的日誌串流不同。

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

您可以在 CloudWatch 主控台上檢查 Apache Flink 的受管理服務量度，以確認應用程式是否正常運作。

清除 AWS 資源

本節包括清除在輪轉視窗教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis 資料串流
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis 資料串流

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇 ExampleOutputStream，選擇動作，選擇刪除，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：在 Python 中建立滑動視窗

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成 入門 (Python) 練習。

本主題包含下列章節：

- 建立相依資源
- 寫入範例記錄至輸入串流
- 下載並檢查應用程式的程式碼
- 壓縮並上傳 Apache Flink 串流 Python 程式碼
- 建立並執行 Managed Service for Apache Flink 應用程式
- 清除 AWS 資源

建立相依資源

在為本練習建立 Managed Service for Apache Flink 應用程式之前，先建立下列相依資源：

- 兩個 Kinesis 資料串流 (`ExampleInputStream` 和 `ExampleOutputStream`)
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (`ka-app-code-<username>`)

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需關於建立這些資源的指示，請參閱以下主題：

- 《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。為資料串流 **`ExampleInputStream`** 和 **`ExampleOutputStream`** 命名。
- 《Amazon Simple Storage Service 使用者指南》中的如何建立 S3 儲存貯體？。透過附加登入名稱 (例如 **`ka-app-code-<username>`**)，為 Amazon S3 儲存貯體提供全域唯一的名稱。

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

> **Note**
>
> 本節中的 Python 指令碼會使用 AWS CLI。您必須將 AWS CLI 設定為使用您的帳戶憑證和預設區域。若要設定 AWS CLI，請輸入下列內容：
>
> ```
> aws configure
> ```

1. 使用下列內容建立名為 `stock.py` 的檔案：

```
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. 執行 stock.py 指令碼：

```
$ python stock.py
```

在完成教學課程的其餘部分時，讓指令碼保持執行狀態。

下載並檢查應用程式的程式碼

此範例的 Python 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 請前往 amazon-kinesis-data-analytics-java-examples/python/SlidingWindow 目錄。

應用程式的程式碼位於 sliding-windows.py 檔案中。請留意下列與應用程式的程式碼相關的資訊：

- 應用程式使用 Kinesis 資料表來源從來源串流讀取。下列程式碼片段會呼叫 create_input_table 函數來建立 Kinesis 資料表來源：

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

create_input_table 函數使用 SQL 命令來建立由串流來源支援的資料表：

```
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)
 }
```

- 應用程式會使用 Slide 運算子來彙總指定滑動視窗內的記錄，並將彙總記錄作為資料表物件傳回：

```
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")
    )
```

- 應用程式會使用 Kinesis Flink 連接器，從 flink-sql-connector-kinesis- 1.15.2.jar 檔案。

壓縮並上傳 Apache Flink 串流 Python 程式碼

在本節中，您會將應用程式的程式碼上傳至在建立相依資源一節建立的 Amazon S3 儲存貯體。

本節說明如何封裝 Python 應用程式。

1. 使用偏好的壓縮應用程式來壓縮 `sliding-windows.py` 和 `flink-sql-connector-kinesis-1.15.2.jar` 檔案。命名存檔 `myapp.zip`。
2. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
3. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `myapp.zip` 檔案。
4. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 針對執行時間，選擇 Apache Flink。

> **Note**
>
> Managed Service for Apache Flink 使用 Apache Flink 1.15.2 版。

- 將版本下拉式清單保留為 Apache Flink 1.15.2 版 (建議版本)。

4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：kinesis-analytics-service-*MyApplication*-*us-west-2*
> - 角色：kinesisanalytics-*MyApplication*-*us-west-2*

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **myapp.zip**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在屬性下，選擇新增群組。
5. 輸入以下應用程式屬性和數值：

群組 ID	金鑰	值
consumer.config.0	**input.stream.name**	**ExampleInputStream**

群組 ID	金鑰	值
consumer.config.0	aws.region	us-west-2
consumer.config.0	scan.stream.initpos	LATEST

選擇儲存。

6. 在屬性下，再次選擇新增群組。
7. 輸入以下應用程式屬性和數值：

群組 ID	金鑰	值
producer.config.0	output.stream.name	ExampleOutputStream
producer.config.0	aws.region	us-west-2
producer.config.0	shard.count	1

8. 在屬性下，再次選擇新增群組。針對群組 ID，輸入 **kinesis.analytics.flink.run.options**。這個特殊的屬性群組會告訴您的應用程式在何處尋找其程式碼資源。如需詳細資訊，請參閱 指定程式碼檔案。
9. 輸入以下應用程式屬性和數值：

群組 ID	金鑰	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. 在監控下，確保監控指標層級設為應用程式。
11. 對於CloudWatch 記錄，請選取啟用核取方塊。
12. 選擇更新。

> **Note**
>
> 當您選擇啟用 CloudWatch 記錄時，Apache Flink 的受管理服務會為您建立記錄群組和記錄資料流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`
>
> 此日誌串流用於監控應用程式。這與應用程式用來傳送結果的日誌串流不同。

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

您可以在 CloudWatch 主控台上檢查 Apache Flink 的受管理服務量度，以確認應用程式是否正常運作。

清除 AWS 資源

本節包括清除在「滑動視窗」教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis 資料串流
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis 資料串流

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇 ExampleOutputStream，選擇動作，選擇刪除，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：使用 Python 將串流資料傳送至 Amazon S3

在本練習中，您將建立適用於 Python 的 Managed Service for Apache Flink 應用程式，將資料串流傳輸到 Amazon Simple Storage Service 接收器。

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成 入門 (Python) 練習。

本主題包含下列章節：

- 建立相依資源
- 寫入範例記錄至輸入串流
- 下載並檢查應用程式的程式碼
- 壓縮並上傳 Apache Flink 串流 Python 程式碼

- 建立並執行 Managed Service for Apache Flink 應用程式
- 清除 AWS 資源

建立相依資源

在為本練習建立 Managed Service for Apache Flink 應用程式之前，先建立下列相依資源：

- Kinesis 資料串流 (`ExampleInputStream`)
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (`ka-app-code-<username>`)

> **Note**
>
> Managed Service for Apache Flink 無法在其自身啟用伺服器端加密的情況下將資料寫入 Amazon S3。

您可以在主控台中建立 Kinesis 串流和 Amazon S3 儲存貯體。如需關於建立這些資源的指示，請參閱以下主題：

- 《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。為資料串流 **`ExampleInputStream`** 命名。
- 《Amazon Simple Storage Service 使用者指南》中的如何建立 S3 儲存貯體？。透過附加登入名稱 (例如 **`ka-app-code-<username>`**)，為 Amazon S3 儲存貯體提供全域唯一的名稱。

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

> **Note**
>
> 本節中的 Python 指令碼會使用 AWS CLI。您必須將 AWS CLI 設定為使用您的帳戶憑證和預設區域。若要設定 AWS CLI，請輸入下列內容：

```
aws configure
```

1. 使用下列內容建立名為 stock.py 的檔案：

```
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. 執行 stock.py 指令碼：

```
$ python stock.py
```

在完成教學課程的其餘部分時，讓指令碼保持執行狀態。

下載並檢查應用程式的程式碼

此範例的 Python 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 請前往 amazon-kinesis-data-analytics-java-examples/python/S3Sink 目錄。

應用程式的程式碼位於 streaming-file-sink.py 檔案中。請留意下列與應用程式的程式碼相關的資訊：

- 應用程式使用 Kinesis 資料表來源從來源串流讀取。下列程式碼片段會呼叫 create_source_table 函數來建立 Kinesis 資料表來源：

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

  create_source_table 函數使用 SQL 命令來建立由串流來源支援的資料表

```
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'))
```

- 應用程式會使用 `filesystem` 連接器將記錄傳送至 Amazon S3 儲存貯體：

```
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'='json',
                  'sink.partition-commit.policy.kind'='success-file',
                  'sink.partition-commit.delay' = '1 min'
              ) """.format(table_name, bucket_name)
```

- 應用程式會使用 Kinesis Flink 連接器，從 flink-sql-connector-kinesis- 1.15.2.jar 檔案。

壓縮並上傳 Apache Flink 串流 Python 程式碼

在本節中，您會將應用程式的程式碼上傳至在建立相依資源一節建立的 Amazon S3 儲存貯體。

1. 使用您偏好的壓縮應用程式來壓縮`streaming-file-sink.py`和 flink-sql-connector-kinesis-1.15.2. jar 檔案。命名存檔 `myapp.zip`。
2. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
3. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `myapp.zip` 檔案。
4. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 針對執行時間，選擇 Apache Flink。

     > **Note**
     >
     > Managed Service for Apache Flink 使用 Apache Flink 1.15.2 版。

   - 將版本下拉式清單保留為 Apache Flink 1.15.2 版 (建議版本)。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：`kinesis-analytics-service-MyApplication-us-west-2`
> - 角色：`kinesisanalytics-MyApplication-us-west-2`

設定應用程式

1. 在MyApplication頁面上，選擇設定。

2. 在設定應用程式頁面，提供程式碼位置：

   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **myapp.zip**。

3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在屬性下，選擇新增群組。
5. 輸入以下應用程式屬性和數值：

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

   選擇儲存。

6. 在屬性下，再次選擇新增群組。針對群組 ID，輸入 **kinesis.analytics.flink.run.options**。這個特殊的屬性群組會告訴您的應用程式在何處尋找其程式碼資源。如需詳細資訊，請參閱 指定程式碼檔案。
7. 輸入以下應用程式屬性和數值：

群組 ID	金鑰	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. 在屬性下，再次選擇新增群組。針對群組 ID，輸入 **sink.config.0**。這個特殊的屬性群組會告訴您的應用程式在何處尋找其程式碼資源。如需詳細資訊，請參閱 指定程式碼檔案。
9. 輸入以下應用程式屬性和值：(以 Amazon S3 儲存貯體的實際名稱取代 *bucket-name*。)

群組 ID	金鑰	值
sink.config.0	**output.bucket.name**	***bucket-name***

10. 在監控下，確保監控指標層級設為應用程式。
11. 對於CloudWatch 記錄，請選取啟用核取方塊。
12. 選擇更新。

> **Note**
>
> 當您選擇啟用 CloudWatch 記錄時，Apache Flink 的受管理服務會為您建立記錄群組和記錄資料流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`
>
> 此日誌串流用於監控應用程式。這與應用程式用來傳送結果的日誌串流不同。

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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>/*"
            ]
        }
    ]
}
```

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

您可以在 CloudWatch 主控台上檢查 Apache Flink 的受管理服務量度，以確認應用程式是否正常運作。

清除 AWS 資源

本節包括清除在「滑動視窗」教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis 資料串流
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis 資料串流

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

Scala 範例

下列範例示範如何搭配使用 Scala 與 Apache Flink 來建立應用程式。

主題

範例：在 Scala 中建立輪轉視窗

> **Note**
>
> 從 Flink 1.15 版開始，移除了 Scala 相依性。應用程式現在可以使用任何 Scala 版本的 Java API。Flink 仍然在內部的幾個關鍵元件中使用 Scala，但不會將 Scala 公開給使用者程式碼類別加載器。因此，使用者需要將 Scala 相依項添加到他們的 jar 存檔中。
> 如需 Flink 1.15 中的 Scala 變更之詳細資訊，請參閱在 1.15 版中移除了 Scala 相依性。

在本練習中，您將創建一個簡單的流應用程序，它使用斯卡拉 3.2.0 和 Flink 的 Java DataStream API。應用程式會從 Kinesis 串流讀取資料，使用滑動視窗彙總資料，並將結果寫入輸出 Kinesis 串流。

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成入門 (Scala) 練習。

本主題包含下列章節：

下載並檢查應用程式的程式碼

此範例的 Python 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 請前往 amazon-kinesis-data-analytics-java-examples/scala/TumblingWindow 目錄。

請留意下列與應用程式的程式碼相關的資訊：

- build.sbt 檔案包含應用程式的組態和相依性資訊，包括 Managed Service for Apache Flink 程式庫。
- BasicStreamingJob.scala 檔案包含定義應用程式功能的主要方法。
- 應用程式使用 Kinesis 來源從來源串流讀取。以下程式碼片段會建立 Kinesis 來源：

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

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

應用程式也會使用 Kinesis 接收器寫入結果串流。以下程式碼片段會建立 Kinesis 目的地：

```
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
}
```

- 該應用程式使用視窗運算子在 5 秒的輪轉視窗內尋找每個股票程式碼的值計數。下列程式碼會建立運算子，並將彙總的資料傳送至新的 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)
```

- 應用程式會建立來源和接收器連接器，以使用 StreamExecutionEnvironment 物件存取外部資源。
- 應用程式會使用動態應用程式屬性來建立來源與目的地連接器。會讀取執行時間應用程式的屬性，來設定連接器。如需執行時間屬性的詳細資訊，請參閱執行時間屬性。

編譯和上傳應用程式的程式碼

在本節中，您會編譯並上傳您應用程式的程式碼至 Amazon S3 儲存貯體。

編譯應用程式的程式碼

使用 SBT 建置工具為應用程式建置 Scala 程式碼。若要安裝 SBT，請參閱使用 cs 安裝程式安裝 sbt。您還需要安裝 Java 開發套件 (JDK)。請參閱完成練習的先決條件。

1. 請將應用程式的程式碼編譯並封裝成 JAR 檔案，以使用應用程式的程式碼。您可以使用 SBT 編譯和封裝程式碼：

```
sbt assembly
```

2. 如果應用程式成功編譯，則會建立下列檔案：

```
target/scala-3.2.0/tumbling-window-scala-1.0.jar
```

上傳 Apache Flink 串流 Scala 程式碼

在本節中，您會建立 Amazon S3 儲存貯體並上傳您應用程式的程式碼。

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 選擇 建立儲存貯體
3. 在儲存貯體名稱欄位中，輸入 `ka-app-code-<username>`。新增尾碼至儲存貯體名稱，例如您的使用者名稱，使其成為全域唯一的。選擇下一步。
4. 在設定選項中，保留原有設定並選擇下一步。
5. 在設定許可步驟中，保留原有設定並選擇下一步。
6. 選擇建立儲存貯體。
7. 選擇 `ka-app-code-<username>` 儲存貯體，然後選擇上傳。
8. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `tumbling-window-scala-1.0.jar` 檔案。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行應用程式 (主控台)

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **`MyApplication`**。
   - 對於 Description (說明)，輸入 **`My Scala test app`**。
   - 針對執行時間，選擇 Apache Flink。
   - 將版本保留為 Apache Flink 版本 1.15.2（建議版本）。
4. 對於存取許可，選擇建立/更新 IAM 角色 **`kinesis-analytics-MyApplication-us-west-2`**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：kinesis-analytics-service-*MyApplication*-*us-west-2*
> - 角色：kinesisanalytics-*MyApplication*-*us-west-2*

設定應用程式

請使用下列程序設定應用程式。

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **tumbling-window-scala-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在屬性下，選擇新增群組。
5. 輸入下列資料：

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

選擇儲存。

6. 在屬性下，再次選擇新增群組。
7. 輸入下列資料：

群組 ID	金鑰	值
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. 在監控下，確保監控指標層級設為應用程式。
9. 對於CloudWatch 記錄，請選擇啟用核取方塊。
10. 選擇更新。

> **Note**
>
> 當您選擇啟用 Amazon 日誌 CloudWatch 記錄時，Apache Flink 的受管服務會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`

編輯 IAM 政策

編輯 IAM 政策以新增 Amazon S3 儲存貯體存取許可。

編輯 IAM 政策以新增 Amazon S3 儲存貯體許可

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。

4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

停止應用程式

若要停止應用程式，請在MyApplication頁面上選擇 [停止]。確認動作。

建立並執行應用程式 (CLI)

在本節中，您可以使用 AWS Command Line Interface 建立和執行 Managed Service for Apache Flink 應用程式。使用 KinesisIntictictics v2 AWS CLI 命令建立 Managed Service for Apache Flink 應用程式並與之互動。

建立許可政策

> **Note**
>
> 您必須為應用程式建立許可政策和角色。如果您未建立這些 IAM 資源，應用程式將無法存取其資料和日誌串流。

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上讀取動作的許可，另一條則是授與目的地串流上寫入動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

使用以下程式碼來建立 AKReadSourceStreamWriteSinkStream 許可政策。以您用於建立 Amazon S3 儲存貯體 (以儲存應用程式的程式碼) 的使用者名稱來取代 **username**。以您的帳戶 ID 取代 Amazon Resource Name (ARN) (**(012345678901)**) 中的帳戶 ID。**MF-stream-rw-role** 服務執行角色應根據客戶特定角色而打造。

```
{
    "ApplicationName": "tumbling_window",
    "ApplicationDescription": "Scala tumbling window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-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"
      }
   ]
}
```

如需建立許可政策的指 step-by-step 示，請參閱 IAM 使用指南中的教學課程：建立和附加您的第一個客戶受管政策。

建立 IAM 角色

在本節中，您會建立 Managed Service for Apache Flink 應用程式可以擔任的 IAM 角色，以便讀取來源串流與寫入目的地串流。

Managed Service for Apache Flink 沒有許可，無法存取串流。您可以透過 IAM 角色來授與這些許可。各 IAM 角色都有連接兩項政策。信任政策會授與擔任角色的 Managed Service for Apache Flink 許可，而許可政策決定了 Managed Service for Apache Flink 在擔任角色後可以執行的作業。

您會將在上一節中建立的許可政策連接至此角色。

若要建立一個 IAM 角色

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽窗格中，選擇角色，然後選擇建立角色。
3. 在選取可信身分類型下，選擇 AWS 服務。
4. 在 Choose the service that will use this role (選擇將使用此角色的服務) 下，選擇 Kinesis。
5. 在選取使用案例下，選擇 Managed Service for Apache Flink。
6. 選擇 Next: Permissions (下一步：許可)。
7. 在 Attach permissions policies (連接許可政策) 頁面上，選擇 Next: Review (下一步：檢閱)。您會在建立角色後連接許可政策。
8. 在建立角色頁面，輸入 **MF-stream-rw-role** 作為角色名稱。選擇建立角色。

   現在您已建立新的 IAM 角色，名為 `MF-stream-rw-role`。您接著會更新角色的信任和許可政策。
9. 將 許可政策連接到角色。

> **Note**
>
> 在此練習中，Managed Service for Apache Flink 擔任從 Kinesis 資料串流 (來源) 讀取資料並將輸出寫入另一個 Kinesis 資料串流的角色。因此您會連接在上一個步驟建立許可政策中建立的政策。

a. 在摘要頁面上，選擇許可標籤。
b. 選擇 Attach Policies (連接政策)。
c. 在搜尋方塊中，輸入 **AKReadSourceStreamWriteSinkStream** (您在上一節中建立的政策)。
d. 選擇 AKReadSourceStreamWriteSinkStream 政策，然後選擇連接政策。

您現在已建立應用程式用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的指 step-by-step 示，請參閱 IAM 使用者指南中的建立 IAM 角色 (主控台)。

建立應用程式

將下列 JSON 程式碼複製到名為 create_request.json 的檔案。以您之前建立之角色的 ARN，取代範例角色 ARN。以您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (username)。以您的帳戶 ID 取代服務執行角色中的範例帳戶 ID (012345678901)。ServiceExecutionRole 應包括您在上一節建立的 IAM 使用者角色。

```
"ApplicationName": "tumbling_window",
    "ApplicationDescription": "Scala getting started application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-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使用以下請求執行以創建應用程序：

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

現在已建立應用程式。您會在下一個步驟中啟動應用程式。

啟動應用程式

在本節中，您會透過 StartApplication 動作來啟動應用程式。

啟動應用程式

1. 將下列 JSON 程式碼複製到名為 start_request.json 的檔案。

```
{
    "ApplicationName": "tumbling_window",
```

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

2. 以啟動應用程式的上述請求，執行 StartApplication 動作：

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

應用程式現在正在執行。您可以在 Amazon CloudWatch 主控台上查看 Apache Flink 的受管服務指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 StopApplication 動作來停止應用程式。

停止應用程式

1. 將下列 JSON 程式碼複製到名為 stop_request.json 的檔案。

```
{
   "ApplicationName": "tumbling_window"
}
```

2. 使用前述請求執行 StopApplication 動作以停止應用程式：

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

現在已停止應用程式。

新增記 CloudWatch 錄選項

您可以使用AWS CLI將 Amazon CloudWatch 日誌串流新增至您的應用程式。如需搭配應用程式使用記 CloudWatch 錄的相關資訊，請參閱設定應用程式記錄。

更新環境屬性

在本節中，您可以使用 UpdateApplication 動作來變更應用程式的環境屬性，無需重新編譯應用程式的程式碼。在此範例中，您會變更來源和目的地串流的「區域」。

更新應用程式的環境屬性

1. 將下列 JSON 程式碼複製到名為 update_properties_request.json 的檔案。

```
{"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. 使用前述請求執行 UpdateApplication 動作以更新環境屬性：

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

更新應用程式的程式碼

當您需要使用新版程式碼套件更新應用程式程式碼時，請使用 UpdateApplicationCLI 動作。

> **Note**
>
> 若要載入具有相同檔案名稱的新版應用程式的程式碼，必須指定新的物件版本。如需關於使用 Amazon S3 物件版本的詳細資訊，請參閱啟用或停用版本控制。

若要使用 AWS CLI，請從 Amazon S3 儲存貯體刪除先前的程式碼套件，上傳新版本，然後呼叫 `UpdateApplication`，指定相同的 Amazon S3 儲存貯體和物件名稱，以及新的物件版本。應用程式將以新的程式碼套件重新啟動。

`UpdateApplication` 動作的下列範例請求會重新載入應用程式的程式碼並重新啟動應用程式。將 `CurrentApplicationVersionId` 更新至目前的應用程式版本。您可以使用 `ListApplications` 或 `DescribeApplication` 動作來檢查目前的應用程式版本。使用您在建立相依資源一節中選擇的尾碼更新儲存貯體名稱尾碼 (<username>)。

```
{
    "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"
                }
            }
        }
    }
}
```

清除 AWS 資源

本節包括清除在「輪轉視窗」教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis 資料串流
- 刪除 Amazon S3 物件與儲存貯體

- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis 資料串流

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇 ExampleOutputStream，選擇動作，選擇刪除，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：在 Scala 中建立滑動視窗

> **Note**
>
> 從 Flink 1.15 版開始，移除了 Scala 相依性。應用程式現在可以使用任何 Scala 版本的 Java API。Flink 仍然在內部的幾個關鍵元件中使用 Scala，但不會將 Scala 公開給使用者程式碼類別加載器。因此，使用者需要將 Scala 相依項添加到他們的 jar 存檔中。
> 如需 Flink 1.15 中的 Scala 變更之詳細資訊，請參閱在 1.15 版中移除了 Scala 相依性。

在本練習中，您將創建一個簡單的流應用程序，它使用斯卡拉 3.2.0 和 Flink 的 Java DataStream API。應用程式會從 Kinesis 串流讀取資料，使用滑動視窗彙總資料，並將結果寫入輸出 Kinesis 串流。

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成入門 (Scala) 練習。

本主題包含下列章節：

- 下載並檢查應用程式的程式碼
- 編譯和上傳應用程式的程式碼
- 建立並執行應用程式 (主控台)
- 建立並執行應用程式 (CLI)
- 更新應用程式的程式碼
- 清除 AWS 資源

下載並檢查應用程式的程式碼

此範例的 Python 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 請前往 amazon-kinesis-data-analytics-java-examples/scala/SlidingWindow 目錄。

請留意下列與應用程式的程式碼相關的資訊：

- build.sbt 檔案包含應用程式的組態和相依性資訊，包括 Managed Service for Apache Flink 程式庫。
- BasicStreamingJob.scala 檔案包含定義應用程式功能的主要方法。
- 應用程式使用 Kinesis 來源從來源串流讀取。以下程式碼片段會建立 Kinesis 來源：

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

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

應用程式也會使用 Kinesis 接收器寫入結果串流。以下程式碼片段會建立 Kinesis 目的地：

```
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
}
```

- 該應用程序使用視窗運算子在按 5 秒滑動的 10 秒視窗內尋找每個股票代碼的值計數。下列程式碼會建立運算子，並將彙總的資料傳送至新的 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)
```

- 應用程式會建立來源和接收器連接器，以使用 StreamExecutionEnvironment 物件存取外部資源。
- 應用程式會使用動態應用程式屬性來建立來源與目的地連接器。會讀取執行時間應用程式的屬性，來設定連接器。如需執行時間屬性的詳細資訊，請參閱執行時間屬性。

編譯和上傳應用程式的程式碼

在本節中，您會編譯並上傳您應用程式的程式碼至 Amazon S3 儲存貯體。

編譯應用程式的程式碼

使用 SBT 建置工具為應用程式建置 Scala 程式碼。若要安裝 SBT，請參閱使用 cs 安裝程式安裝 sbt。您還需要安裝 Java 開發套件 (JDK)。請參閱完成練習的先決條件。

1. 請將應用程式的程式碼編譯並封裝成 JAR 檔案，以使用應用程式的程式碼。您可以使用 SBT 編譯和封裝程式碼：

```
sbt assembly
```

2. 如果應用程式成功編譯，則會建立下列檔案：

```
target/scala-3.2.0/sliding-window-scala-1.0.jar
```

上傳 Apache Flink 串流 Scala 程式碼

在本節中，您會建立 Amazon S3 儲存貯體並上傳您應用程式的程式碼。

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 選擇 建立儲存貯體
3. 在儲存貯體名稱欄位中，輸入 `ka-app-code-<username>`。新增尾碼至儲存貯體名稱，例如您的使用者名稱，使其成為全域唯一的。選擇下一步。
4. 在設定選項中，保留原有設定並選擇下一步。
5. 在設定許可步驟中，保留原有設定並選擇下一步。
6. 選擇建立儲存貯體。
7. 選擇 `ka-app-code-<username>` 儲存貯體，然後選擇上傳。
8. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `sliding-window-scala-1.0.jar` 檔案。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行應用程式 (主控台)

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **`MyApplication`**。
   - 對於 Description (說明)，輸入 **`My Scala test app`**。
   - 針對執行時間，選擇 Apache Flink。
   - 將版本保留為 Apache Flink 版本 1.15.2（建議版本）。
4. 對於存取許可，選擇建立/更新 IAM 角色 **`kinesis-analytics-MyApplication-us-west-2`**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：kinesis-analytics-service-*MyApplication*-*us-west-2*
> - 角色：kinesisanalytics-*MyApplication*-*us-west-2*

設定應用程式

請使用下列程序設定應用程式。

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **sliding-window-scala-1.0.jar.**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在屬性下，選擇新增群組。
5. 輸入下列資料：

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

選擇儲存。

6. 在屬性下，再次選擇新增群組。
7. 輸入下列資料：

群組 ID	金鑰	值
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. 在監控下，確保監控指標層級設為應用程式。
9. 對於CloudWatch 記錄，請選擇啟用核取方塊。
10. 選擇更新。

> **Note**
>
> 當您選擇啟用 Amazon 日誌 CloudWatch 記錄時，Apache Flink 的受管服務會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`

編輯 IAM 政策

編輯 IAM 政策以新增 Amazon S3 儲存貯體存取許可。

編輯 IAM 政策以新增 Amazon S3 儲存貯體許可

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。

4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

停止應用程式

若要停止應用程式，請在MyApplication頁面上選擇 [停止]。確認動作。

建立並執行應用程式 (CLI)

在本節中，您可以使用 AWS Command Line Interface 建立和執行 Managed Service for Apache Flink 應用程式。使用 KinesisIntictictics v2 AWS CLI 命令建立 Managed Service for Apache Flink 應用程式並與之互動。

建立許可政策

> **Note**
>
> 您必須為應用程式建立許可政策和角色。如果您未建立這些 IAM 資源，應用程式將無法存取其資料和日誌串流。

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上讀取動作的許可，另一條則是授與目的地串流上寫入動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

使用以下程式碼來建立 AKReadSourceStreamWriteSinkStream 許可政策。以您用於建立 Amazon S3 儲存貯體 (以儲存應用程式的程式碼) 的使用者名稱來取代 **username**。以您的帳戶 ID 取代 Amazon Resource Name (ARN) (**(012345678901)**) 中的帳戶 ID。

```
{
    "ApplicationName": "sliding_window",
    "ApplicationDescription": "Scala sliding window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-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"
      }
   ]
}
```

如需建立許可政策的指 step-by-step 示，請參閱 IAM 使用指南中的教學課程：建立和附加您的第一個客戶受管政策。

建立 IAM 角色

在本節中，您會建立 Managed Service for Apache Flink 應用程式可以擔任的 IAM 角色，以便讀取來源串流與寫入目的地串流。

Managed Service for Apache Flink 沒有許可，無法存取串流。您可以透過 IAM 角色來授與這些許可。各 IAM 角色都有連接兩項政策。信任政策會授與擔任角色的 Managed Service for Apache Flink 許可，而許可政策決定了 Managed Service for Apache Flink 在擔任角色後可以執行的作業。

您會將在上一節中建立的許可政策連接至此角色。

若要建立一個 IAM 角色

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽窗格中，選擇角色，然後選擇建立角色。
3. 在選取可信身分類型下，選擇 AWS 服務。
4. 在 Choose the service that will use this role (選擇將使用此角色的服務) 下，選擇 Kinesis。
5. 在選取使用案例下，選擇 Managed Service for Apache Flink。
6. 選擇 Next: Permissions (下一步：許可)。
7. 在 Attach permissions policies (連接許可政策) 頁面上，選擇 Next: Review (下一步：檢閱)。您會在建立角色後連接許可政策。
8. 在建立角色頁面，輸入 **MF-stream-rw-role** 作為角色名稱。選擇建立角色。

   現在您已建立新的 IAM 角色，名為 `MF-stream-rw-role`。您接著會更新角色的信任和許可政策。
9. 將 許可政策連接到角色。

> **Note**
>
> 在此練習中，Managed Service for Apache Flink 擔任從 Kinesis 資料串流 (來源) 讀取資料並將輸出寫入另一個 Kinesis 資料串流的角色。因此您會連接在上一個步驟建立許可政策中建立的政策。

a. 在摘要頁面上，選擇許可標籤。
b. 選擇 Attach Policies (連接政策)。
c. 在搜尋方塊中，輸入 **AKReadSourceStreamWriteSinkStream** (您在上一節中建立的政策)。
d. 選擇 AKReadSourceStreamWriteSinkStream 政策，然後選擇連接政策。

您現在已建立應用程式用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的指 step-by-step 示，請參閱 IAM 使用者指南中的建立 IAM 角色 (主控台)。

建立應用程式

將下列 JSON 程式碼複製到名為 create_request.json 的檔案。以您之前建立之角色的 ARN，取代範例角色 ARN。以您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (username)。以您的帳戶 ID 取代服務執行角色中的範例帳戶 ID (012345678901)。

```
{
    "ApplicationName": "sliding_window",
    "ApplicationDescription": "Scala sliding_window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-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使用以下請求執行以創建應用程序：

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

現在已建立應用程式。您會在下一個步驟中啟動應用程式。

啟動應用程式

在本節中，您會透過 StartApplication 動作來啟動應用程式。

啟動應用程式

1. 將下列 JSON 程式碼複製到名為 `start_request.json` 的檔案。

```
{
    "ApplicationName": "sliding_window",
```

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

2. 以啟動應用程式的上述請求，執行 StartApplication 動作：

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

應用程式現在正在執行。您可以在 Amazon CloudWatch 主控台上查看 Apache Flink 的受管服務指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 StopApplication 動作來停止應用程式。

停止應用程式

1. 將下列 JSON 程式碼複製到名為 stop_request.json 的檔案。

```
{
   "ApplicationName": "sliding_window"
}
```

2. 使用前述請求執行 StopApplication 動作以停止應用程式：

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

現在已停止應用程式。

新增記 CloudWatch 錄選項

您可以使用AWS CLI將 Amazon CloudWatch 日誌串流新增至您的應用程式。如需搭配應用程式使用記 CloudWatch 錄的相關資訊，請參閱設定應用程式記錄。

更新環境屬性

在本節中，您可以使用 UpdateApplication 動作來變更應用程式的環境屬性，無需重新編譯應用程式的程式碼。在此範例中，您會變更來源和目的地串流的「區域」。

更新應用程式的環境屬性

1. 將下列 JSON 程式碼複製到名為 update_properties_request.json 的檔案。

```
{"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. 使用前述請求執行 UpdateApplication 動作以更新環境屬性：

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

更新應用程式的程式碼

當您需要使用新版程式碼套件更新應用程式程式碼時，請使用 UpdateApplicationCLI 動作。

> **Note**
>
> 若要載入具有相同檔案名稱的新版應用程式的程式碼，必須指定新的物件版本。如需關於使用 Amazon S3 物件版本的詳細資訊，請參閱啟用或停用版本控制。

若要使用 AWS CLI，請從 Amazon S3 儲存貯體刪除先前的程式碼套件，上傳新版本，然後呼叫 `UpdateApplication`，指定相同的 Amazon S3 儲存貯體和物件名稱，以及新的物件版本。應用程式將以新的程式碼套件重新啟動。

`UpdateApplication` 動作的下列範例請求會重新載入應用程式的程式碼並重新啟動應用程式。將 `CurrentApplicationVersionId` 更新至目前的應用程式版本。您可以使用 `ListApplications` 或 `DescribeApplication` 動作來檢查目前的應用程式版本。使用您在建立相依資源一節中選擇的尾碼更新儲存貯體名稱尾碼 (<username>)。

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

清除 AWS 資源

本節包括清除在滑動視窗教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis 資料串流
- 刪除 Amazon S3 物件與儲存貯體

- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis 資料串流

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇 ExampleOutputStream，選擇動作，選擇刪除，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

範例：使用 Scala 將串流資料傳送至 Amazon S3

> **Note**
>
> 從 Flink 1.15 版開始，移除了 Scala 相依性。應用程式現在可以使用任何 Scala 版本的 Java API。Flink 仍然在內部的幾個關鍵元件中使用 Scala，但不會將 Scala 公開給使用者程式碼類別加載器。因此，使用者需要將 Scala 相依項添加到他們的 jar 存檔中。
> 如需 Flink 1.15 中的 Scala 變更之詳細資訊，請參閱在 1.15 版中移除了 Scala 相依性。

在本練習中，您將創建一個簡單的流應用程序，它使用斯卡拉 3.2.0 和 Flink 的 Java DataStream API。應用程式會從 Kinesis 串流讀取資料，使用滑動視窗彙總資料，並將結果寫入 S3。

> **Note**
>
> 若要設定此練習的必要先決條件，請先完成入門 (Scala) 練習。您只需要在 Amazon S3 存儲桶**data/**中創建一個額外的文件夾 ka-app-code- <username>。

本主題包含下列章節：

- 下載並檢查應用程式的程式碼
- 編譯和上傳應用程式的程式碼
- 建立並執行應用程式 (主控台)
- 建立並執行應用程式 (CLI)
- 更新應用程式的程式碼
- 清除 AWS 資源

下載並檢查應用程式的程式碼

此範例的 Python 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，請執行下列動作：

1. 如果您尚未安裝 Git 用戶端，請先完成安裝。如需詳細資訊，請參閱安裝 Git。
2. 使用以下指令複製遠端儲存庫：

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

3. 請前往 amazon-kinesis-data-analytics-java-examples/scala/S3Sink 目錄。

請留意下列與應用程式的程式碼相關的資訊：

- build.sbt 檔案包含應用程式的組態和相依性資訊，包括 Managed Service for Apache Flink 程式庫。
- BasicStreamingJob.scala 檔案包含定義應用程式功能的主要方法。
- 應用程式使用 Kinesis 來源從來源串流讀取。以下程式碼片段會建立 Kinesis 來源：

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

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

該應用程序還使 StreamingFileSink 用一個寫入 Amazon S3 存儲桶：`

```
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()
}
```

- 應用程式會建立來源和接收器連接器，以使用 StreamExecutionEnvironment 物件存取外部資源。
- 應用程式會使用動態應用程式屬性來建立來源與目的地連接器。會讀取執行時間應用程式的屬性，來設定連接器。如需執行時間屬性的詳細資訊，請參閱執行時間屬性。

編譯和上傳應用程式的程式碼

在本節中，您會編譯並上傳您應用程式的程式碼至 Amazon S3 儲存貯體。

編譯應用程式的程式碼

使用 SBT 建置工具為應用程式建置 Scala 程式碼。若要安裝 SBT，請參閱使用 cs 安裝程式安裝 sbt。您還需要安裝 Java 開發套件 (JDK)。請參閱完成練習的先決條件。

1. 請將應用程式的程式碼編譯並封裝成 JAR 檔案，以使用應用程式的程式碼。您可以使用 SBT 編譯和封裝程式碼：

```
sbt assembly
```

2. 如果應用程式成功編譯，則會建立下列檔案：

```
target/scala-3.2.0/s3-sink-scala-1.0.jar
```

上傳 Apache Flink 串流 Scala 程式碼

在本節中，您會建立 Amazon S3 儲存貯體並上傳您應用程式的程式碼。

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 選擇 建立儲存貯體
3. 在儲存貯體名稱欄位中，輸入 ka-app-code-<username>。新增尾碼至儲存貯體名稱，例如您的使用者名稱，使其成為全域唯一的。選擇下一步。
4. 在設定選項中，保留原有設定並選擇下一步。
5. 在設定許可步驟中，保留原有設定並選擇下一步。
6. 選擇建立儲存貯體。
7. 選擇 ka-app-code-<username> 儲存貯體，然後選擇上傳。
8. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 s3-sink-scala-1.0.jar 檔案。

9. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行應用程式 (主控台)

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 對於 Description (說明)，輸入 **My java test app**。
   - 針對執行時間，選擇 Apache Flink。
   - 將版本保留為 Apache Flink 版本 1.15.2（建議版本）。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - 角色：`kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

設定應用程式

請使用下列程序設定應用程式。

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **s3-sink-scala-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在屬性下，選擇新增群組。
5. 輸入下列資料：

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

   選擇儲存。
6. 在屬性下，選擇新增群組。
7. 輸入下列資料：

群組 ID	金鑰	值
ProducerConfigProperties	**s3.sink.path**	**s3a://ka-app-code-*<user-name>* /data**

8. 在監控下，確保監控指標層級設為應用程式。
9. 對於CloudWatch 記錄，請選擇啟用核取方塊。
10. 選擇更新。

> **Note**
>
> 當您選擇啟用 Amazon 日誌 CloudWatch 記錄時，Apache Flink 的受管服務會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`

編輯 IAM 政策

編輯 IAM 政策以新增 Amazon S3 儲存貯體存取許可。

編輯 IAM 政策以新增 Amazon S3 儲存貯體許可

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

停止應用程式

若要停止應用程式，請在MyApplication頁面上選擇 [停止]。確認動作。

建立並執行應用程式 (CLI)

在本節中，您可以使用 AWS Command Line Interface 建立和執行 Managed Service for Apache Flink 應用程式。使用 KinesisIntictictics v2 AWS CLI 命令建立 Managed Service for Apache Flink 應用程式並與之互動。

建立許可政策

> **Note**
>
> 您必須為應用程式建立許可政策和角色。如果您未建立這些 IAM 資源，應用程式將無法存取其資料和日誌串流。

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上讀取動作的許可，另一條則是授與目的地串流上寫入動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

使用以下程式碼來建立 AKReadSourceStreamWriteSinkStream 許可政策。以您用於建立 Amazon S3 儲存貯體 (以儲存應用程式的程式碼) 的使用者名稱來取代 **username**。以您的帳戶 ID 取代 Amazon Resource Name (ARN) (**(012345678901)**) 中的帳戶 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"
        }
    ]
}
```

如需建立許可政策的指 step-by-step 示，請參閱 IAM 使用指南中的教學課程：建立和附加您的第一個客戶受管政策。

建立 IAM 角色

在本節中，您會建立 Managed Service for Apache Flink 應用程式可以擔任的 IAM 角色，以便讀取來源串流與寫入目的地串流。

Managed Service for Apache Flink 沒有許可，無法存取串流。您可以透過 IAM 角色來授與這些許可。各 IAM 角色都有連接兩項政策。信任政策會授與擔任角色的 Managed Service for Apache Flink 許可，而許可政策決定了 Managed Service for Apache Flink 在擔任角色後可以執行的作業。

您會將在上一節中建立的許可政策連接至此角色。

若要建立一個 IAM 角色

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽窗格中，選擇角色，然後選擇建立角色。
3. 在選取可信身分類型下，選擇 AWS 服務。
4. 在 Choose the service that will use this role (選擇將使用此角色的服務) 下，選擇 Kinesis。
5. 在選取使用案例下，選擇 Managed Service for Apache Flink。
6. 選擇 Next: Permissions (下一步：許可)。
7. 在 Attach permissions policies (連接許可政策) 頁面上，選擇 Next: Review (下一步：檢閱)。您會在建立角色後連接許可政策。
8. 在建立角色頁面，輸入 **`MF-stream-rw-role`** 作為角色名稱。選擇建立角色。

   現在您已建立新的 IAM 角色，名為 `MF-stream-rw-role`。您接著會更新角色的信任和許可政策。
9. 將 許可政策連接到角色。

> **Note**
>
> 在此練習中，Managed Service for Apache Flink 擔任從 Kinesis 資料串流 (來源) 讀取資料並將輸出寫入另一個 Kinesis 資料串流的角色。因此您會連接在上一個步驟建立許可政策中建立的政策。

a. 在摘要頁面上，選擇許可標籤。
b. 選擇 Attach Policies (連接政策)。
c. 在搜尋方塊中，輸入 **AKReadSourceStreamWriteSinkStream** (您在上一節中建立的政策)。
d. 選擇 AKReadSourceStreamWriteSinkStream 政策，然後選擇連接政策。

您現在已建立應用程式用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的指 step-by-step 示，請參閱 IAM 使用者指南中的建立 IAM 角色 (主控台)。

建立應用程式

將下列 JSON 程式碼複製到名為 create_request.json 的檔案。以您之前建立之角色的 ARN，取代範例角色 ARN。以您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (username)。以您的帳戶 ID 取代服務執行角色中的範例帳戶 ID (012345678901)。

```
{
    "ApplicationName": "s3_sink",
    "ApplicationDescription": "Scala tumbling window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-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使用以下請求執行以創建應用程序：

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

現在已建立應用程式。您會在下一個步驟中啟動應用程式。

啟動應用程式

在本節中，您會透過 StartApplication 動作來啟動應用程式。

啟動應用程式

1. 將下列 JSON 程式碼複製到名為 `start_request.json` 的檔案。

```
{{
    "ApplicationName": "s3_sink",
    "RunConfiguration": {
```

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

2. 以啟動應用程式的上述請求，執行 StartApplication 動作：

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

應用程式現在正在執行。您可以在 Amazon CloudWatch 主控台上查看 Apache Flink 的受管服務指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 StopApplication 動作來停止應用程式。

停止應用程式

1. 將下列 JSON 程式碼複製到名為 stop_request.json 的檔案。

```
{
   "ApplicationName": "s3_sink"
}
```

2. 使用前述請求執行 StopApplication 動作以停止應用程式：

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

現在已停止應用程式。

新增記 CloudWatch 錄選項

您可以使用AWS CLI將 Amazon CloudWatch 日誌串流新增至您的應用程式。如需搭配應用程式使用記 CloudWatch 錄的相關資訊，請參閱設定應用程式記錄。

更新環境屬性

在本節中，您可以使用 UpdateApplication 動作來變更應用程式的環境屬性，無需重新編譯應用程式的程式碼。在此範例中，您會變更來源和目的地串流的「區域」。

更新應用程式的環境屬性

1. 將下列 JSON 程式碼複製到名為 update_properties_request.json 的檔案。

```
{"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. 使用前述請求執行 UpdateApplication 動作以更新環境屬性：

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

更新應用程式的程式碼

當您需要使用新版程式碼套件更新應用程式程式碼時，請使用 UpdateApplicationCLI 動作。

> **Note**
>
> 若要載入具有相同檔案名稱的新版應用程式的程式碼，必須指定新的物件版本。如需關於使用 Amazon S3 物件版本的詳細資訊，請參閱啟用或停用版本控制。

若要使用 AWS CLI，請從 Amazon S3 儲存貯體刪除先前的程式碼套件，上傳新版本，然後呼叫 UpdateApplication，指定相同的 Amazon S3 儲存貯體和物件名稱，以及新的物件版本。應用程式將以新的程式碼套件重新啟動。

UpdateApplication 動作的下列範例請求會重新載入應用程式的程式碼並重新啟動應用程式。將 CurrentApplicationVersionId 更新至目前的應用程式版本。您可以使用 ListApplications 或 DescribeApplication 動作來檢查目前的應用程式版本。使用您在建立相依資源一節中選擇的尾碼更新儲存貯體名稱尾碼 (<username>)。

```
{
    "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"
                }
            }
        }
    }
}
```

清除 AWS 資源

本節包括清除在輪轉視窗教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis 資料串流
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台

2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis 資料串流

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇 ExampleOutputStream，選擇動作，選擇刪除，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

Amazon Managed Service for Apache Flink 中的安全性

雲端安全是 AWS 最重視的一環。身為 AWS 的客戶，您將能從資料中心和網路架構中獲益，這些都是專為最重視安全的組織而設計的。

安全是 AWS 與您共同的責任。共同的責任模型 將此描述為雲端 本身 的安全和雲端內部的安全：

- 雲端本身的安全：AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。AWS 也提供您可安全使用的服務。第三方稽核人員定期檢測及驗證安全的效率也是我們 AWS 合規計劃的一部分。若要了解適用於 Managed Service for Apache Flink 的合規計劃，請參閱合規計劃範圍內的 AWS 服務。
- 雲端內部的安全：您的責任取決於所使用的 AWS 服務。您也必須對資料敏感度、組織要求，以及適用法律和法規等其他因素負責。

本文件有助於您了解如何在使用 Managed Service for Apache Flink 時套用共同責任模式。以下主題說明如何設定 Managed Service for Apache Flink 以符合您的安全和合規目標。您也將了解如何使用其他 Amazon 服務，協助監控並保護 Managed Service for Apache Flink 資源。

主題

Amazon Managed Service for Apache Flink 中的資料保護

您可以使用 AWS 提供的工具來保護您的資料。Managed Service for Apache Flink 可與支援資料加密的服務搭配使用，包括 Kinesis Data Firehose 和 Amazon S3。

Managed Service for Apache Flink 中的資料加密

靜態加密

請注意以下有關使用 Managed Service for Apache Flink 加密靜態資料的相關事項：

- 您可以使StartStreamEncryption用加密傳入 Kinesis 資料串流上的資料。如需詳細資訊，請參閱什麼是 Kinesis Data Streams 伺服器端加密？。
- 輸出資料可以使用 Kinesis Data Firehose 進行靜態加密，以將該資料存放在加密的 Amazon S3 儲存貯體中。您可以指定 Amazon S3 儲存貯體使用的加密金鑰。如需詳細資訊，請參閱搭配使用伺服器端加密與 KMS 受管金鑰 (SSE-KMS) 來保護資料。
- Managed Service for Apache Flink 可從任何串流來源讀取，並可寫入任何串流或資料庫目的地。確保您的來源和目的地會加密所有傳輸中的資料和靜態資料。
- 應用程式的程式碼採用靜態加密。
- 耐久性應用程式儲存體採用靜態加密。
- 執行中的應用程式儲存體採用靜態加密。

傳輸中加密

Managed Service for Apache Flink 會將所有傳輸中的資料加密。所有 Managed Service for Apache Flink 應用程式都會啟用傳輸中的加密，且無法停用。

Managed Service for Apache Flink 會在下列情況下加密傳輸中的資料：

- 資料從 Kinesis Data Streams 傳輸到 Managed Service for Apache Flink。
- 資料在 Managed Service for Apache Flink 內部元件間傳輸。
- 資料在 Managed Service for Apache Flink 和 Kinesis Data Streams 間傳輸。

金鑰管理

Managed Service for Apache Flink 中的資料加密使用服務受管金鑰。不支援客戶受管金鑰。

Amazon Managed Service for Apache Flink 的身分和存取管理

AWS Identity and Access Management (IAM) 是一種 AWS 服務，讓管理員能夠安全地控制對 AWS 資源的存取權。IAM 管理員可以控制誰能完成身分驗證 (登入) 和獲得授權 (具有許可)，而得以使用 Managed Service for Apache Flink 資源。IAM 是一種您可以免費使用的 AWS 服務。

主題

- 物件

- 使用身分驗證
- 使用政策管理存取權
- Amazon Managed Service for Apache Flink 如何與 IAM 搭配使用
- Amazon Managed Service for Apache Flink 的身分型政策範例
- Amazon Managed Service for Apache Flink 的身分和存取權疑難排解
- 預防跨服務混淆代理人

物件

AWS Identity and Access Management (IAM) 的使用方式會有所不同，取決於您在 Managed Service for Apache Flink 中所執行的工作。

服務使用者：如果使用 Managed Service for Apache Flink 服務執行工作，管理員會為您提供所需的憑證和許可。隨著您為了進行工作而使用更多的 Managed Service for Apache Flink 功能，您可能會需要額外的許可。瞭解存取許可的管理方式可協助您向管理員請求正確的許可。若您無法存取 Managed Service for Apache Flink 中的某項功能，請參閱 Amazon Managed Service for Apache Flink 的身分和存取權疑難排解。

服務管理員：如果您負責公司內的 Managed Service for Apache Flink 資源，您可能具備 Managed Service for Apache Flink 的完整存取權。您的任務是判斷服務使用者應該存取的 Managed Service for Apache Flink 功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司可搭配 Managed Service for Apache Flink 使用 IAM 的方式，請參閱 Amazon Managed Service for Apache Flink 如何與 IAM 搭配使用。

IAM 管理員：如果您是 IAM 管理員，建議您掌握如何編寫政策的詳細資訊，以管理 Managed Service for Apache Flink 存取權。如需檢視您可以在 IAM 中使用的 Apache Flink 身分型政策範例，請參閱 Amazon Managed Service for Apache Flink 的身分型政策範例。

使用身分驗證

身分驗證是使用身分憑證登入 AWS 的方式。您必須以 AWS 帳戶根使用者、IAM 使用者身分，或擔任 IAM 角色進行 驗證 (登入至 AWS)。

您可以使用透過身分來源 AWS IAM Identity Center 提供的憑證，以聯合身分登入 AWS。(IAM Identity Center) 使用者、貴公司的單一登入身分驗證和您的 Google 或 Facebook 憑證都是聯合身分的範例。

您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。您 AWS 藉由使用聯合進行存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入至 AWS 的詳細資訊，請參閱《AWS 登入 使用者指南》中的如何登入您的 AWS 帳戶。

如果您是以程式設計的方式存取 AWS，AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以便使用您的憑證透過密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，您必須自行簽署請求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱《IAM 使用者指南》中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如，AWS 建議您使用多重要素驗證 (MFA) 以提高帳戶的安全。如需更多資訊，請參閱《AWS IAM Identity Center 使用者指南》中的多重要素驗證和《IAM 使用者指南》中的在 AWS 中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

如果是建立 AWS 帳戶，您會先有一個登入身分，可以完整存取帳戶中所有 AWS 服務 與資源。此身分稱為 AWS 帳戶 根使用者，使用建立帳戶時所使用的電子郵件地址和密碼即可登入並存取。強烈建議您不要以根使用者處理日常作業。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

聯合身分

最佳實務是要求人類使用者 (包括需要管理員存取權的使用者)搭配身分提供者使用聯合功能，使用暫時憑證來存取 AWS 服務。

聯合身分是來自您企業使用者目錄的使用者、Web 身分供應商、AWS Directory Service、Identity Center 目錄或透過身分來源提供的憑證來存取 AWS 服務 的任何使用者。聯合身分存取 AWS 帳戶時，會擔任角色，並由角色提供暫時憑證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分來源中的一組使用者和群組，以便在您的所有 AWS 帳戶和應用程式中使用。如需 IAM Identity Center 的相關資訊，請參閱《AWS IAM Identity Center 使用者指南》中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是您 AWS 帳戶 中的一種身分，具備單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰)的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需詳細資訊，請參閱《IAM 使用者指

南》https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#rotate-credentials中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分登入。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的過程變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱《IAM 使用者指南》中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 中的一種身分，具備特定許可。它類似 IAM 使用者，但不與特定的人員相關聯。您可以在 AWS Management Console 中透過切換角色來暫時取得 IAM 角色。您可以透過呼叫 AWS CLI 或 AWS API 操作，或是使用自訂 URL 來取得角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並取得由角色定義的許可。如需有關聯合角色的詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-idp.html中的為第三方身分供應商建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱《AWS IAM Identity Center 使用者指南》中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳户存取權 – 您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的委託人)存取您帳戶中的資源。角色是授予跨帳户存取權的主要方式。但是，針對某些 AWS 服務，您可以將政策直接連接到資源 (而非使用角色作為代理)。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源類型政策的差異。
- 跨服務存取 – 有些 AWS 服務 會使用其他 AWS 服務 中的功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉發存取工作階段 (FAS)：當您使用 IAM 使用者或角色在 AWS 中執行動作時，系統會將您視為主體。當您使用某些服務時，您可能會執行一個動作，而該動作之後會在不同的服務中啟動另一個

動作。FAS 使用主體的許可呼叫 AWS 服務，搭配請求 AWS 服務 以向下游服務發出請求。只有在服務收到需要與其他 AWS 服務 或資源互動才能完成的請求之後，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

- 服務角色：服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可給 AWS 服務 服務。
- 服務連結角色 – 服務連結角色是一種連結到 AWS 服務 的服務角色類型。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 AWS 帳戶 中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

- 在 Amazon EC2 上執行的應用程式 – 針對在 EC2 執行個體上執行並提出 AWS CLI 和 AWS API 請求的應用程式，您可以使用 IAM 角色來管理暫時憑證。這是在 EC2 執行個體內儲存存取金鑰的較好方式。如需指派 AWS 角色給 EC2 執行個體並提供其所有應用程式使用，您可以建立連接到執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱《IAM 使用者指南》中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透過建立政策並將其附加到 AWS 身分或資源，在 AWS 中控制存取。政策是 AWS 中的一個物件，當其和身分或資源建立關聯時，便可定義其許可。AWS 會在主體 (使用者、根使用者或角色工作階段)發出請求時評估這些政策。政策中的許可，決定是否允許或拒絕請求。大部分政策以 JSON 文件形式儲存在 AWS 中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱《IAM 使用者指南》中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授與使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具備該政策的使用者便可以從 AWS Management Console、AWS CLI 或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色)的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策則是獨立的政策，您可以將這些政策附加到 AWS 帳戶 中的多個使用者、群組和角色。受管政策包含 AWS 管理政策和客戶管理政策。如需瞭解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主體可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人 (帳戶成員、使用者或角色)擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon Simple Storage Service (Amazon S3)、AWS WAF 和 Amazon VPC 是支援 ACL 的服務範例。若要進一步了解 ACL，請參閱《Amazon Simple Storage Service 開發人員指南》中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較少見的政策類型。這些政策類型可設定較常見政策類型授與您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色)的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可範圍的更多相關資訊，請參閱《IAM 使用者指南》中的 IAM 實體許可範圍。

- 服務控制政策 (SCP) – SCP 是 JSON 政策，可指定 AWS Organizations 中組織或組織單位 (OU) 的最大許可。AWS Organizations 服務可用來分組和集中管理您企業所擁有的多個 AWS 帳戶。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個 AWS 帳戶根使用者。如需組織和 SCP 的更多相關資訊，請參閱《AWS Organizations 使用者指南》中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需更多資訊，請參閱《IAM 使用者指南》中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。如需瞭解 AWS 在涉及多種政策類型時如何判斷是否允許一項請求，請參閱 IAM 使用者指南中的政策評估邏輯。

Amazon Managed Service for Apache Flink 如何與 IAM 搭配使用

在使用 IAM 管理 Managed Service for Apache Flink 的存取權之前，應先了解可以搭配 Managed Service for Apache Flink 使用的 IAM 功能有哪些。

您可以搭配 Amazon Managed Service for Apache Flink 使用的 IAM 功能

IAM 功能	Managed Service for Apache Flink 支援
身分型政策	是
資源型政策	否
政策動作	是
政策資源	是
政策條件索引鍵	否
ACL	否
ABAC (政策中的標籤)	是
臨時憑證	是

IAM 功能	Managed Service for Apache Flink 支援
主體許可	是
服務角色	否
服務連結角色	否

如要全面了解 Managed Service for Apache Flink 和其他 AWS 服務如何與大多數的 IAM 功能搭配使用，請參閱《IAM 使用者指南》中的可搭配 IAM 運作的 AWS 服務。

Managed Service for Apache Flink 內的身分型政策

支援身分型政策	是

身分型政策是可以連接到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要瞭解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至附加的使用者或角色。如要瞭解您在 JSON 政策中使用的所有元素，請參閱 IAM 使用者指南中的 IAM JSON 政策元素參考。

Managed Service for Apache Flink 內的身分型政策範例

如需檢視 Managed Service for Apache Flink 身分型政策範例，請參閱 Amazon Managed Service for Apache Flink 的身分型政策範例。

Managed Service for Apache Flink 內的資源型政策

支援以資源基礎的政策	是

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執

行的動作。您必須在資源型政策中指定主體。主體可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

若要啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源在不同的 AWS 帳戶 中時，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 存取資源的許可。其透過將身分型政策附加到實體來授予許可。不過，如果資源型政策會為相同帳戶中的主體授與存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM 角色與資源型政策有何差異。

Managed Service for Apache Flink 的政策動作

支援政策動作	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。政策動作的名稱通常會和相關聯的 AWS API 操作相同。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些操作需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授與執行相關聯操作的許可。

若要查看 Managed Service for Apache Flink 動作的清單，請參閱《服務授權參考》中的 Amazon Managed Service for Apache Flink 定義的動作。

Managed Service for Apache Flink 中的政策動作會在動作之前使用以下字首：

```
Kinesis Analytics
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "Kinesis Analytics:action1",
      "Kinesis Analytics:action2"
         ]
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 Describe 文字的所有動作，請包含以下動作：

```
"Action": "Kinesis Analytics:Describe*"
```

若要檢視 Managed Service for Apache Flink 以身分為基礎的政策範例，請參閱。Amazon Managed Service for Apache Flink 的身分型政策範例

適用於 Managed Service for Apache Flink 的政策資源

支援政策資源	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出作業)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

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

若要查看 Managed Service for Apache Flink 資源類型及其 ARN 的詳細資訊，請參閱服務授權參考中的 Amazon Managed Service for Apache Flink 定義的資源。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon Managed Service for Apache Flink 定義的動作。

若要檢視 Managed Service for Apache Flink 身分型政策的範例，請參閱。Amazon Managed Service for Apache Flink 的身分型政策範例

適用於 Managed Service for Apache Flink 的政策條件索引鍵

支援服務特定政策條件索引鍵	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊)可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵，AWS 會使用邏輯 AND 操作評估他們。若您為單一條件索引鍵指定多個值，AWS 會使用邏輯 OR 操作評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授與該 IAM 使用者。如需更多資訊，請參閱《IAM 使用者指南》中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件索引鍵和服務特定的條件索引鍵。若要查看 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的 AWS 全域條件內容索引鍵。

若要查看 Managed Service for Apache Flink 條件索引鍵的清單，請參閱《服務授權參考》中的 Amazon Managed Service for Apache Flink 的條件索引鍵。若要了解您可以搭配哪些動作和資源使用條件金鑰，請參閱 Amazon Managed Service for Apache Flink 定義的動作。

若要檢視 Managed Service for Apache Flink 以身分為基礎的政策範例，請參閱。Amazon Managed Service for Apache Flink 的身分型政策範例

Managed Service for Apache Flink 中的存取控制清單 (ACL)

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

屬性型存取控制 (ABAC)用於 Managed Service for Apache Flink

支援 ABAC (政策中的標籤)	是

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在 AWS 中，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色)，以及許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

若要根據標籤控制存取，請使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件索引鍵，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件索引鍵，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的使用屬性型存取控制 (ABAC)。

將臨時憑證用於 Managed Service for Apache Flink

支援臨時憑證	是

您使用臨時憑證進行登入時，某些 AWS 服務 無法運作。如需詳細資訊，包括那些 AWS 服務 搭配臨時憑證運作，請參閱《IAM 使用者指南》中的可搭配 IAM 運作的 AWS 服務。

如果您使用使用者名稱和密碼之外的任何方法登入 AWS Management Console，則您正在使用臨時憑證。例如，當您使用公司的單一登入(SSO)連結存取 AWS 時，該程序會自動建立臨時憑證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱 IAM 使用者指南中的切換至角色 (主控台)。

您可使用 AWS CLI 或 AWS API，手動建立臨時憑證。接著，您可以使用這些臨時憑證來存取 AWS。AWS 建議您動態產生臨時憑證，而非使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

Managed Service for Apache Flink 的跨服務主體許可

支援轉寄存取工作階段 (FAS)	是

當您使用 IAM 使用者或角色在 AWS 中執行動作時，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用主體的許可呼叫 AWS 服務，搭配請求 AWS 服務 以向下游服務發出請求。只有在服務收到需要與其他 AWS 服務 或資源互動才能完成的請求之後，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

Managed Service for Apache Flink 的服務角色

支援服務角色	是

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可給 AWS 服務 服務。

> **Warning**
>
> 變更服務角色的許可有可能會中止 Managed Service for Apache Flink 的功能。僅當 Managed Service for Apache Flink 有提供相關指引時，才能編輯服務角色。

適用於 Managed Service for Apache Flink 的服務連結角色

支援服務連結角色	是

服務連結角色是一種連結到 AWS 服務 的服務角色類型。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 AWS 帳戶 中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理服務連結角色的詳細資訊，請參閱可搭配 IAM 運作的 AWS 服務。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

Amazon Managed Service for Apache Flink 的身分型政策範例

根據預設，使用者和角色不具備建立或修改 Managed Service for Apache Flink 資源的許可。他們也無法使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 或 AWS API 執行任務。若要授與使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

如需 Managed Service for Apache Flink 所定義之動作和資源類型的詳細資訊，包括每種資源類型的 ARN 格式，請參閱《服務授權參考》中的 Amazon Managed Service for Apache Flink 適用的動作、資源和條件索引鍵。

主題

政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 Managed Service for Apache Flink 資源。這些動作可能會讓您的 AWS 帳戶 產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管政策並朝向最低權限許可的目標邁進：如需開始授予許可給使用者和工作負載，請使用 AWS 受管政策，這些政策會授予許可給許多常用案例。它們可在您的 AWS 帳戶 中使用。我們建議您定義特定於使用案例的 AWS 客戶管理政策，以便進一步減少許可。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的 AWS 受管政策。
- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授予對服務動作的存取權，前提是透過特定 AWS 服務 (例如 AWS CloudFormation)使用條件。如需更多資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多重要素驗證 (MFA)：如果存在需要 AWS 帳戶中 IAM 使用者或根使用者的情況，請開啟 MFA 提供額外的安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

有關 IAM 中最佳實務的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 最佳安全實務。

使用 Managed Service in Apache Flink 主控台

若要存取 Amazon Managed Service for Apache Flink 主控台，您必須擁有最基本的一組許可。這些許可必須可讓您列出和檢視您 AWS 帳戶 帳戶中 Managed Service for Apache Flink 資源的詳細資訊。如

果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許其最基本主控台許可。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

為確保使用者和角色仍可使用 Managed Service for Apache Flink 主控台，還要將 Managed Service for Apache Flink `ConsoleAccess` 或 `ReadOnly` AWS 受管政策連接至實體。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此政策包含在主控台上，或是使用 AWS CLI 或 AWS API 透過編寫程式的方式完成此動作的許可。

```
{
    "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": "*"
        }
    ]
}
```

Amazon Managed Service for Apache Flink 的身分和存取權疑難排解

請使用以下資訊來協助診斷和修正使用 Managed Service for Apache Flink 和 IAM 時可能遇到的常見問題。

主題

- 我未獲授權，無法在 Managed Service for Apache Flink 中執行動作
- 我沒有授權執行 iam：PassRole
- 我想要允許 AWS 帳戶以外的人員存取我的 Managed Service for Apache Flink 資源

我未獲授權，無法在 Managed Service for Apache Flink 中執行動作

若 AWS Management Console 告知您並未獲得執行動作的授權，您必須聯絡您的管理員以取得協助。您的管理員是提供您使用者名稱和密碼的人員。

下列範例錯誤會在 `mateojackson` 使用者嘗試使用主控台檢視一個虛構 *`my-example-widget`* 資源的詳細資訊，但卻無虛構 `Kinesis Analytics:`*`GetWidget`* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: Kinesis
 Analytics:GetWidget on resource: my-example-widget
```

在此情況下，Mateo 會請求管理員更新他的政策，允許他使用 *`my-example-widget`* 動作存取 `Kinesis Analytics:`*`GetWidget`* 資源。

我沒有授權執行 iam：PassRole

如果錯誤訊息告知您未獲得授權，無法執行 `iam:PassRole` 動作，您的政策就必須更新，以允許將角色傳遞給 Managed Service for Apache Flink。

有些 AWS 服務 允許您傳遞現有的角色至該服務，而無須建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

名為 marymajor 的 IAM 使用者嘗試使用主控台在 Managed Service for Apache Flink 中執行動作時，會發生下列範例所示的錯誤。但是，該動作要求服務具備服務角色授與的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如需任何協助，請聯絡您的 AWS 管理員。您的管理員提供您的登入憑證。

我想要允許 AWS 帳戶以外的人員存取我的 Managed Service for Apache Flink 資源

您可以建立一個角色，讓其他帳戶中的使用者或您的組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 Managed Service for Apache Flink 是否支援這些功能，請參閱 Amazon Managed Service for Apache Flink 如何與 IAM 搭配使用。
- 如需了解如何存取您擁有的所有 AWS 帳戶 所提供的資源，請參閱《IAM 使用者指南》中的將存取權提供給您所擁有的另一個 AWS 帳戶 中的 IAM 使用者。
- 如需了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的將存取權提供給第三方擁有的 AWS 帳戶。
- 如需了解如何透過聯合身分提供存取權，請參閱《IAM 使用者指南》中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源型政策的差異。

預防跨服務混淆代理人

在 AWS 中，當某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，會發生跨服務模擬。可以操縱呼叫服務來對其他客戶的資源採取動作，即使該服務不應有適當的許可，導致混淆代理人的問題。

為了預防混淆代理人的情況，AWS 提供了工具來協助您保護所有服務的資料，而這些服務的主體已獲得您帳戶中資源的存取權。本節重點介紹特定於 Managed Service for Apache Flink 的防範跨服務混淆代理人；但是，您可以在《IAM 使用者指南》的混淆代理人問題一節，了解此主題的更多資訊。

在 Apache Flink 受管服務的內容中，我們建議在角色信任政策中使用 aws: SourceArn 和 aws: SourceAccount 全域條件上下文金鑰，將角色的存取限制為只有預期資源產生的請求。

如果您想要僅允許一個資源與跨服務存取相關聯，則請使用 aws:SourceArn。如果您想要允許該帳戶中的任何資源與跨服務使用相關聯，請使用 aws:SourceAccount。

aws:SourceArn 的值必須是 Managed Service for Apache Flink 所使用之資源的 ARN，其格式指定如下：arn:aws:kinesisanalytics:region:account:resource

防範混淆代理人問題的建議方法是使用 aws:SourceArn 全域條件內容索引鍵以及資源的完整 ARN。

如果不知道資源的完整 ARN，或者如果您指定了多個資源，請使用 aws:SourceArn 鍵搭配萬用字元 (*) 來表示 ARN 的未知部分。例如：arn:aws:kinesisanalytics::111122223333:*。

您提供給 Managed Service for Apache Flink 的角色政策，以及為您產生之角色的信任政策，都可以使用這些索引鍵。

請執行下列步驟以防範混淆代理人問題：

防範混淆代理人問題

1. 登入 AWS 管理主控台，然後前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 請選擇角色，然後選擇您要修改的角色。
3. 選擇編輯信任政策。
4. 在編輯信任原則頁面上，使用一個或兩個 aws:SourceArn 和 aws:SourceAccount 全域條件內容索引鍵的政策取代預設 JSON 政策。請參閱以下政策範例：
5. 選擇 更新政策。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"kinesisanalytics.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"Account ID"
```

```
            },
            "ArnEquals":{
               "aws:SourceArn":"arn:aws:kinesisanalytics:us-
east-1:123456789012:application/my-app"
            }
         }
      }
   ]
}
```

監控 Managed Service for Apache Flink

Managed Service for Apache Flink 可為應用程式提供監控功能。如需詳細資訊，請參閱 記錄和監控。

Amazon Managed Service for Apache Flink 的合規驗證

在多個 AWS 合規計劃中，第三方稽核機構會評估 Amazon Managed Service for Apache Flink 的安全與合規。這些包括 SOC、PCI、HIPAA 等。

如需特定合規計劃範圍內的 AWS 服務清單，請參閱：如需一般資訊，請參閱 AWS 合規計劃。

您可使用 AWS Artifact 下載第三方稽核報告。如需詳細資訊，請參閱在 AWS Artifact 中下載報告。

您使用 Managed Service for Apache Flink 時的合規責任，取決於資料的機密性、您公司的合規目標及適用法律和法規。若您使用 Managed Service for Apache Flink 必須遵循特定標準，如 HIPAA 或 PCI，AWS 會提供資源予以協助：

- 安全與合規快速入門指南 – 這些部署指南討論在 AWS 上部署以安全及合規為重心基準環境的架構考量和步驟。
- Amazon Web Services 上的 HIPAA 安全與合規架構。本白皮書說明公司可如何運用 AWS 來建立 HIPAA 合規的應用程式。
- AWS 合規資源 – 這組手冊和指南可能適用於您的產業和位置。
- AWS Config – 此 AWS 服務可評定資源組態與內部實務、業界準則和法規的合規狀態。
- AWS Security Hub：此 AWS 服務可供您檢視 AWS 中的安全狀態，可助您檢查是否符合安全產業標準和最佳實務。

FedRAMP

AWS FedRAMP 合規計劃把 Managed Service for Apache Flink 納為 FedRAMP 授權服務。如果您是聯邦或商業客戶，您可以使用此服務在 AWS GovCloud (美國) 區域的授權邊界處理和儲存敏感工作負載，其資料可達到高影響層級，以及資料高達中等層級的美國東部 (維吉尼亞北部)、美國東部 (俄亥俄)、美國西部 (加利佛尼亞北部)、美國西部 (奧勒岡) 區域。

您可以透過 FedRAMP PMO 或向 AWS 銷售客戶經理請求存取 AWS FedRAMP 安全套件，或藉由 AWS Artifact 中的 AWS Artifact 來下載套件。

如需詳細資訊，請參閱 FedRAMP。

Amazon Managed Service for Apache Flink 中的彈性

AWS 全球基礎設施是以 AWS 區域與可用區域為中心建置的。AWS區域提供多個分開且隔離的實際可用區域，它們以低延遲、高輸送量和高度備援聯網功能相互連結。透過可用區域，您所設計與操作的應用程式和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需有關 AWS 區域與可用區域的更多相關資訊，請參閱 AWS 全球基礎設施。

除 AWS 全球基礎設施外， Managed Service for Apache Flink 另亦提供數種功能，可協助支援資料的復原能力和備份需求。

災難復原

Managed Service for Apache Flink 會在無伺服器模式中執行，並透過執行自動遷移，來處理主機降級、可用區域可用性和其他基礎設施相關的問題。Managed Service for Apache Flink 透過多重備援機制來實現這一目標。每個 Managed Service for Apache Flink 應用程式都會在單一租用戶 Apache Flink 叢集中執行。Apache Flink 叢集與 JobMananger 在高可用性模式下使用動物園管理員跨多個可用區域執行。Managed Service for Apache Flink 使用 Amazon EKS 部署 Apache Flink。在 Amazon EKS 中針對跨可用區域的每個 AWS 區域使用了多個 Kubernetes Pod。萬一發生故障，Managed Service for Apache Flink 會先嘗試使用應用程式的檢查點 (如果有) 復原執行中 Apache Flink 叢集內的應用程式。

Managed Service for Apache Flink 使用檢查點和快照備份應用程式狀態：

- 檢查點是應用程式狀態的備份，Managed Service for Apache Flink 會定期自動建立這些狀態，並用來從錯誤中還原。
- 快照是您手動建立和還原的應用程式狀態備份。

如需檢查點和快照的詳細資訊，請參閱容錯能力。

版本控制

應用程式狀態的存儲版本的版本控制如下：

- 檢查點由服務自動建立版本。如果服務使用檢查點重新啟動應用程式，則會使用最新的檢查點。
- 儲存點是使用動作SnapshotName參數進行版本化的。CreateApplicationSnapshot

Managed Service for Apache Flink 會對儲存在檢查點和儲存點中的資料進行加密。

Managed Service for Apache Flink 中的基礎設施安全性

Managed Service for Apache Flink 作為受管服務，受《Amazon Web Services：安全程序概觀》https://d0.awsstatic.com/whitepapers/Security/AWS_Security_Whitepaper.pdf白皮書所述的 AWS 全球網路安全程序所保護。

您可使用 AWS 發佈的 API 呼叫，透過網路存取 Managed Service for Apache Flink 。對 Managed Service for Apache Flink 的所有 API 呼叫都會透過 Transport Layer Security (TLS) 保護，並透過 IAM 進行驗證。用戶端必須支援 TLS 1.2 或更新版本。用戶端也必須支援具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

Managed Service for Apache Flink 安全最佳實務

在您開發和實作自己的安全政策時，可考慮使用 Amazon Managed Service for Apache Flink 提供的多種安全功能。以下最佳實務為一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

實作最低權限存取

當您授與許可時，需要決定哪些使用者會取得哪些 Apache Flink 資源的哪些許可。您還需針對這些資源啟用允許執行的動作，因此，您只應授與執行任務所需的許可。對降低錯誤或惡意意圖所引起的安全風險和影響而言，實作最低權限存取是相當重要的一環。

使用 IAM 角色存取其他 Amazon 服務

您的 Managed Service for Apache Flink 應用程式必須具有有效的憑證，才能存取其他服務中的資源，例如 Kinesis 資料串流、Kinesis Data Firehose 串流或 Amazon S3 儲存貯體。您不應該將 AWS 憑證直接儲存在應用程式或 Amazon S3 儲存貯體中。這些是不會自動輪換的長期憑證，如果遭到盜用，可能會對業務造成嚴重的影響。

反之，您應使用 IAM 角色為應用程式管理暫時性憑證，以存取其他資源。使用角色時，您不必使用長期憑證來存取其他資源。

如需詳細資訊，請參閱《IAM 使用者指南》中的以下主題：

- IAM 角色
- 常見的角色方案：使用者、應用程式和服務

在相依資源實作伺服器端加密

靜態資料和傳輸中的資料會在 Managed Service for Apache Flink 中加密，而且此加密無法停用。您應該在相依資源 (例如 Kinesis 資料串流、Kinesis Data Firehose 串流和 Amazon S3 儲存貯體) 中實作伺服器端加密。如需關於在相依資源中實作伺服器端加密的詳細資訊，請參閱 資料保護。

用 CloudTrail 於監控 API 呼叫

Managed Service for Apache Flink 已與 AWS CloudTrail 整合，這是一項服務，可提供使用者、角色或 Amazon 服務在 Managed Service for Apache Flink 中所採取動作的記錄。

使用收集的資訊 CloudTrail，您可以判斷對 Apache Flink 的受管理服務提出的要求、提出要求的 IP 位址、提出要求的人員、提出要求的時間，以及其他詳細資訊。

如需詳細資訊，請參閱 the section called “使用 AWS CloudTrail”。

Amazon Managed Service for Apache Flink 的記錄和監控

監控是維護 Managed Service for Apache Flink 應用程式之可靠性、可用性和效能的重要部分。您應該全面收集 AWS 解決方案的監控資料，以便在發生故障時更容易偵錯。

在開始監控 Managed Service for Apache Flink 之前，應先建立監控規劃，為下列問題提供解答：

- 監控目標是什麼？
- 要監控哪些資源？
- 監控這些資源的頻率為何？
- 要使用哪些監控工具？
- 誰將執行監控任務？
- 發生問題時應該通知誰？

下一個步驟是在您的環境中建立正常 Managed Service for Apache Flink 效能的基準。您可以在各種時間及不同負載條件下測量效能，以完成此項作業。監控 Managed Service for Apache Flink 時，可以儲存歷史監控資料。如此您才能與目前的效能資料做比較、辨識正常效能模式和效能異常狀況和規劃問題處理方式。

主題

日誌

記錄對於生產應用程式了解錯誤和失敗非常重要。不過，記錄子系統需要收集記錄項目並將記錄項目轉寄至記錄 CloudWatch 檔。雖然有些記錄可正常且理想，但大量記錄可能會使服務超載，並造成 Flink 應用程式落後。記錄例外狀況和警告當然是一個好主意。但是您無法為 Flink 應用程式處理的每則訊

息產生日誌訊息。Flink 針對高輸出量和低延遲進行最佳化，但記錄子系統沒有。如果確實需要為每個已處理的消息生成日誌輸出，請在 Flink 應用程序 DataStream 中使用另一個附加信息並使用適當的接收器將數據發送到 Amazon S3 或 CloudWatch。請勿將 Java 記錄系統用於此目的。此外，Managed Service for Apache Flink 的 `Debug Monitoring Log Level` 設定會產生大量流量，從而會造成背壓。您只能在主動調查應用程式問題時使用它。

使用記錄檔見解查詢 CloudWatch 記錄

CloudWatch 日誌見解是一項功能強大的服務，可大規模查詢日誌。客戶應利用其功能快速搜尋日誌，以識別並減輕操作事件期間的錯誤。

下列查詢會在所有任務管理員記錄中尋找例外狀況，並根據其發生的時間排序。

```
fields @timestamp, @message
| filter isPresent(throwableInformation.0) or isPresent(throwableInformation) or
 @message like /(Error|Exception)/
| sort @timestamp desc
```

如需其他有用的查詢，請參閱範例查詢。

監控

在生產環境中執行串流應用程式時，您著手持續且無限期地執行應用程式。它對實現所有元件的監控和適當報警至關重要，不只是對 Flink 應用程式。否則，您可能會較早就錯過新出現的問題，只在問題完全顯現並且更難以緩解之後才意識到操作事件。要監控的一般事項包括：

- 來源是否正在擷取資料？
- 是否已從來源讀取資料 (從來源角度看)？
- Flink 應用程式是否正在接收資料？
- Flink 應用程式能夠跟上還是落後？
- Flink 應用程式是否將資料保存到接收器中 (從應用程式角度看)？
- 接收器是否正在接收資料？

然後應該考慮針對 Flink 應用程式的更具體指標。此CloudWatch 儀表板提供了良好的起點。如需要為生產應用程式監控的指標之詳細資訊，請參閱將 CloudWatch 警示與 Amazon 管理服務搭配 Apache Flink 使用。這些指標包括：

- records_lag_max 和 MillisHindLatest：如果應用程式正在從 Kinesis 或 Kafka 取用，這些指標會指出應用程式是否落後，需要進行縮減以跟上目前的負載。這是一個很好的通用指標，很容易跟踪各種應用程式。但它只能用於被動式擴展，也就是說，當應用程式已經落後時。
- CPU cpuU tilization 和 heapMemoryUtilization— 這些指標可以很好地指示應用程式的整體資源使用率，並且可用於主動擴充，除非應用程式是 I/O 繫結。
- downtime：停機時間大於零表示應用程式失敗。如果值大於 0，表示應用程式未在處理任何資料。
- lastCheckpointSize和 lastCheckpointDuration— 這些指標會監控狀態中儲存的資料量，以及需要多長時間才能取得檢查點。如果檢查點增長或花費較長時間，應用程式會持續花費時間在檢查點上，而且實際處理的週期較少。在某些時候，檢查點可能會變得太大或花費很長時間才會失敗。除了監控絕對值之外，客戶還應考慮使用 `RATE(lastCheckpointSize)` 和 `RATE(lastCheckpointDuration)` 監控變動率。
- numberOfFailed檢查點 — 此測量結果會計算應用程式啟動後失敗的檢查點數目。根據應用程式的不同，如果檢查點偶爾失敗，該指標可以容忍。但是，如果檢查點經常出現故障，則應用程式可能運作不正常，需要進一步關注。我們建議監控 `RATE(numberOfFailedCheckpoints)` 梯度警報，而不是絕對值。

設定應用程式記錄

透過將 Amazon CloudWatch 記錄選項新增至 Apache Flink 應用程式的受管服務，您可以監控應用程式事件或組態問題。

本主題說明如何將應用程式設定為將應用程式事件寫入 CloudWatch 記錄資料流。 CloudWatch 記錄選項是應用程式設定和權限的集合，您的應用程式用來設定將應用程式事件寫入記 CloudWatch 錄的方式。您可以使用AWS Management Console或 AWS Command Line Interface (AWS CLI) 來新增和設定 CloudWatch 記錄選項。

請注意下列有關將 CloudWatch 記錄選項新增至應用程式的事項：

- 當您使用主控台新增 CloudWatch 記錄選項時，Apache Flink 的受管理服務會為您建立記 CloudWatch 錄群組和記錄資料流，並新增應用程式寫入記錄串流所需的權限。
- 使用 API 新增 CloudWatch 記錄選項時，您還必須建立應用程式的記錄群組和記錄資料流，並新增應用程式寫入記錄資料流所需的權限。

本主題包含下列章節：

- 使用主控台設定 CloudWatch 記錄

使用主控台設定 CloudWatch 記錄

當您在主控台中啟用應用程式的 CloudWatch 記錄時，系統會為您建立記錄群組和記錄資料流。CloudWatch 此外，您應用程式的許可政策也會更新為具有寫入串流的許可。

Apache Flink 的受管理服務會建立使用下列慣例命名的記錄群組，其中*ApplicationName*是應用程式的名稱。

```
/AWS/KinesisAnalytics/ApplicationName
```

Managed Service for Apache Flink 會以下列名稱在新日誌群組中建立日誌串流。

```
kinesis-analytics-log-stream
```

您可以使用設定應用程式頁面的監控日誌層級區塊，設定應用程式的監控指標層級和監控日誌層級。如需關於應用程式日誌層級的詳細資訊，請參閱the section called “應用程式監控層級”。

使用 CLI 設定 CloudWatch 記錄

若要使用新增 CloudWatch 記錄選項AWS CLI，請執行下列動作：

- 建立 CloudWatch 記錄群組和記錄資料流。
- 當您使用動作建立應用程式時，新增記錄選項，或使用CreateApplication動作將記錄選項新增至現有的AddApplicationCloudWatchLoggingOption應用程式。
- 將許可新增至應用程式的政策中，以寫入日誌。

本節包含下列主題：

- 使用應用程式 CloudWatch 記錄選項
- 新增寫入 CloudWatch 記錄資料流的權限

建立 CloudWatch 記錄群組和記錄資料流

您可以使用 CloudWatch CloudWatch 記錄主控台或 API 建立記錄群組和串流。如需建立記錄群組和 CloudWatch 記錄資料流的相關資訊，請參閱使用記錄群組和記錄串流。

使用應用程式 CloudWatch 記錄選項

使用下列 API 動作將 CloudWatch 記錄選項新增至新的或現有的應用程式，或變更現有應用程式的記錄選項。如需關於如何使用 JSON 檔案作為 API 動作輸入的資訊，請參閱 Managed Service for Apache Flink API 範例程式碼。

建立應用程式時新增 CloudWatch 記錄選項

下面的例子演示了如何在創建應用程序時使用該`CreateApplication`操作添加 CloudWatch 日誌選項。在此範例中，將 *`CloudWatch ##### Amazon #### (ARN) ################`*。如需這些動作的詳細資訊，請參閱 `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>"
 }]
}
```

將 CloudWatch 記錄選項新增至現有的應用程式

下面的例子演示了如何使用該AddApplicationCloudWatchLoggingOption操作將 CloudWatch log 選項添加到現有的應用程序。在下列範例中，使用您自己的資訊取代每個*********。如需這些動作的詳細資訊，請參閱 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>
}
```

更新現有的 CloudWatch 記錄選項

下面的實例演示瞭如何使用UpdateApplication動作來修改現有的 CloudWatch log 選項。在下列範例中，使用您自己的資訊取代每個*********。如需這些動作的詳細資訊，請參閱 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>
}
```

從應用程式刪除 CloudWatch 記錄選項

下面的實例演示了如何使用該DeleteApplicationCloudWatchLoggingOption操作來刪除現有的 CloudWatch 日誌選項。在下列範例中，使用您自己的資訊取代每個*********。如需這些動作的詳細資訊，請參閱 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>
}
```

設定應用程式記錄層級

若要設定應用程式記錄層級，請使用 CreateApplication 動作的 MonitoringConfiguration 參數或 UpdateApplication 動作的 MonitoringConfigurationUpdate 參數。

如需關於應用程式日誌層級的詳細資訊，請參閱the section called “應用程式監控層級”。

建立應用程式時設定應用程式記錄層級

CreateApplication 動作的下列範例請求會將應用程式日誌層級設定為 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"
}
```

更新應用程式記錄層級

UpdateApplication 動作的下列範例請求會將應用程式日誌層級設定為 INFO。

```
{
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "MonitoringConfigurationUpdate": {
            "ConfigurationTypeUpdate": "CUSTOM",
            "LogLevelUpdate": "INFO"
         }
      }
   }
}
```

新增寫入 CloudWatch 記錄資料流的權限

適用於 Apache Flink 的受管理服務需要將錯誤設定錯誤寫入的權限。 CloudWatch您可以將這些許可新增至 Managed Service for Apache Flink 所承擔的 AWS Identity and Access Management (IAM) 角色。

如需將 IAM 角色用於 Managed Service for Apache Flink 的詳細資訊，請參閱Amazon Managed Service for Apache Flink 的身分和存取管理。

信任政策

若要授與 Managed Service for Apache Flink 許可以擔任 IAM 角色，您可以將以下信任政策附加到服務執行角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanlaytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

許可政策

若要 CloudWatch 從 Apache Flink 的受管理服務資源授與應用程式寫入日誌事件的許可，您可以使用下列 IAM 許可政策。為日誌群組和串流提供正確的 Amazon Resource Name (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:*",
            ]
        }
    ]
}
```

應用程式監控層級

您可以使用應用程式監控指標層級和監控日誌層級來控制應用程式日誌訊息的產生。

應用程式的監控指標層級會控制日誌訊息的精細程度。監控指標層級的定義如下：

- 應用程式：指標的適用範圍是整個應用程式。
- 任務：指標的適用範圍是每個任務。如需任務的相關資訊，請參閱 the section called “擴展”。
- 運算子：指標的適用範圍是每個運算子。如需運算子的相關資訊，請參閱 the section called “DataStream API 運算子”。
- 平行處理層級：指標的適用範圍是應用程式平行處理層級。您只能使用 UpdateApplicationAPI 的 MonitoringConfigurationUpdate參數設定此指標層級。您無法使用控制台來設定此指標層級。如需平行處理層級的相關資訊，請參閱 the section called “擴展”。

應用程式的監控日誌層級會控制應用程式日誌的詳細程度。監控日誌層級的定義如下：

- 錯誤：應用程式的潛在災難性事件。
- 警告：應用程式的可能有害的情況。
- 資訊：應用程式的資訊和暫時性故障事件。建議您使用此記錄層級。
- 偵錯：對於應用程式偵錯最有用的精細資訊事件。注意：此層級僅適用於暫時偵錯之目的。

記錄最佳實務

我們建議您的應用程式使用資訊記錄層級。我們建議您使用此層級以確保看到 Apache Flink 錯誤，這些錯誤記錄在資訊層級而非錯誤層級。

我們建議您只在調查應用程式問題時暫時使用偵錯層級。問題解決後，切換回資訊層級。使用偵錯記錄層級將顯著影響應用程式的效能。

過多的記錄也會大幅影響應用程式效能。例如，建議您不要為每筆處理的記錄寫入日誌項目。過多的記錄可能會導致嚴重的資料處理瓶頸，並可能導致從來源讀取資料時產生背壓。

記錄疑難排解

如果應用程式日誌未寫入日誌串流，請驗證下列項目：

- 確認應用程式的 IAM 角色和政策正確無誤。您的應用程式政策需要下列許可才能存取日誌串流：
  - `logs:PutLogEvents`
  - `logs:DescribeLogGroups`
  - `logs:DescribeLogStreams`

  如需詳細資訊，請參閱 the section called “新增寫入 CloudWatch 記錄資料流的權限”。
- 確認應用程式正在執行。若要檢查應用程式的狀態，請在主控台中檢視應用程式的頁面，或使用DescribeApplication或ListApplications動作。
- 監視 CloudWatch 指標，例如診downtime斷其他應用程式問題。如需有關讀取 CloudWatch 量度的資訊，請參閱Managed Service for Apache Flink 中的指標和維度。

後續步驟

在應用程式中啟用 CloudWatch 記錄之後，您可以使用 CloudWatch 日誌深入解析來分析應用程式記錄檔。如需詳細資訊，請參閱 the section called “分析日誌”。

使用日誌洞察分析 CloudWatch 日誌

如上一節所述，將 CloudWatch 記錄選項新增至應用程式之後，您可以使用 Lo CloudWatch gs Insights 查詢特定事件或錯誤的記錄資料流。

CloudWatch 日誌洞見可讓您以互動方式搜尋和分析 CloudWatch 記錄中的記錄資料。

如需開始使用 CloudWatch 日誌洞見的相關資訊，請參閱使用日誌深入解析分析 CloudWatch 記錄資料。

執行範例查詢

本節說明如何執行範例 CloudWatch 記錄見解查詢。

先決條件

- 在記錄檔中設定的現有記錄群組和 CloudWatch 記錄串流。
- 記錄檔中儲存的現有 CloudWatch 記錄檔。

如果您使用 Amazon 路線 53 或 Amazon VPC 之類的服務，則可能已經從這些服務設置了日誌以轉到 CloudWatch 日誌。AWS CloudTrail如需將記錄檔傳送至 CloudWatch 記錄檔的詳細資訊，請參閱 CloudWatch 記錄檔入門。

CloudWatch Logs Insights 中的查詢會傳回記錄事件中的一組欄位，或是數學彙總或對記錄事件執行的其他作業的結果。本節示範的查詢會傳回日誌事件清單。

若要執行 CloudWatch 日誌見解範例查詢

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Insights。
3. 熒幕上方附近的查詢編輯器包含可傳回 20 筆最新日誌事件的預設查詢。在查詢編輯器上方，選取要查詢的日誌群組。

   當您選取記錄群組時， CloudWatch Logs Insights 會自動偵測記錄群組中資料的欄位，並將這些欄位顯示在右窗格的 [探查] 欄位中。它也會顯示一段時間內此日誌群組中日誌事件的長條圖。此長條圖顯示日誌群組中符合您的查詢和時間範圍的事件分佈，而不只是表格中顯示的事件。
4. 選擇 Run query (執行查詢)。

查詢的結果隨即出現。在這個範例中，結果是最新的 20 個日誌事件 (任何類型)。

5. 若要查看其中一個傳回的日誌事件的所有欄位，請選擇該日誌事件左側的箭頭。

如需如何執行和修改 CloudWatch 記錄見解查詢的相關資訊，請參閱執行和修改查詢範例。

查詢範例

本節包含 CloudWatch 記錄見解範例查詢，用於分析 Apache Flink 應用程式記錄的受管理服務。這些查詢會搜尋數個範例錯誤條件，並作為撰寫查詢以尋找其他錯誤條件的範本。

> **Note**
>
> 將下列查詢範例中的地區 (*#### -2*)、帳戶識別碼 (*012345678901*) 和應用程式名稱 (*YourApplication*) 取代為應用程式的「地區」和「帳戶 ID」。

本主題包含下列章節：

- 分析操作：任務分配
- 分析操作：平行處理層級變更
- 分析錯誤：存取遭拒
- 分析錯誤：找不到來源或接收器
- 分析錯誤：應用程式任務相關的失敗

分析操作：任務分配

下列 CloudWatch 記錄見解查詢會傳回 Apache Flink Job 管理員在工作管理員之間分配的工作數目。您需要將查詢的時間範圍設定為符合一個作業執行，以便查詢不會傳回來自先前作業的任務。如需平行處理層級的詳細資訊，請參閱擴展。

```
fields @timestamp, message
| filter message like /Deploying/
| parse message " to flink-taskmanager-*" as @tmid
| stats count(*) by @tmid
| sort @timestamp desc
| limit 2000
```

下列 CloudWatch 記錄見解查詢會傳回指派給每個工作管理員的子工作。子任務的總數是每個任務的平行處理層級的總和。任務平行處理層級衍生自運算子平行處理層級，且預設會與應用程式的平行處理層級相同，除非您在程式碼中指定 `setParallelism` 來變更它。如需關於設定運算子平行處理層級的詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的設定平行處理層級：運算子層級。

```
fields @timestamp, @tmid, @subtask
| filter message like /Deploying/
| parse message "Deploying * to flink-taskmanager-*" as @subtask, @tmid
| sort @timestamp desc
| limit 2000
```

如需關於任務排程的詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的作業和排程。

分析操作：平行處理層級變更

下列 CloudWatch Logs Insights 查詢會傳回應用程式平行處理原則的變更 (例如，由於自動調整規模)。此查詢還會傳回應用程式平行處理層級的手動變更。如需自動擴展的相關資訊，請參閱the section called “自動擴展”。

```
fields @timestamp, @parallelism
| filter message like /property: parallelism.default, /
| parse message "default, *" as @parallelism
| sort @timestamp asc
```

分析錯誤：存取遭拒

下列 CloudWatch 記錄深入解析查詢會傳回`Access Denied`記錄檔。

```
fields @timestamp, @message, @messageType
| filter applicationARN like /arn:aws:kinesisanalytics us-
west-2:012345678901:application\/YourApplication/
| filter @message like /AccessDenied/
| sort @timestamp desc
```

分析錯誤：找不到來源或接收器

下列 CloudWatch 記錄深入解析查詢會傳回`ResourceNotFound`記錄檔。 `ResourceNotFound`如果找不到 Kinesis 來源或接收器，則記錄結果。

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalytics us-
west-2:012345678901:application\/YourApplication/
| filter @message like /ResourceNotFoundException/
| sort @timestamp desc
```

分析錯誤：應用程式任務相關的失敗

下列 CloudWatch 記錄見解查詢會傳回應用程式的工作相關失敗記錄。如果應用程式的狀態從 RUNNING 切換到 RESTARTING，就會產生這些日誌結果。

```
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
```

對於使用 Apache Flink 1.8.2 版及以前版本的應用程式，任務相關的失敗將導致應用程式狀態反而從 RUNNING 切換到 FAILED。使用 Apache Flink 1.8.2 及之前版本時，請使用下列查詢來搜尋與應用程式任務相關的失敗：

```
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
```

檢視 Managed Service for Apache Flink 中的指標和維度

本主題包含下列章節：

- 將 CloudWatch 警示與 Amazon 管理服務搭配 Apache Flink 使用

當您的 Apache Flink 受管服務處理資料來源時，適用於 Apache Flink 的受管服務會向 Amazon 報告以下指標和維度。 CloudWatch

應用程式指標

指標	單位	描述	Level	使用須知
backPressuredTimeMsPerSecond*	毫秒	此任務或運算子每秒承受背壓的時間 (毫秒)。	任務、運算子、平行處理層級	* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。 這些指標可用於識別應用程式中的瓶頸。
busyTimeMsPerSecond*	毫秒	此任務或運算子每秒忙碌 (既不是閒置也沒有背壓) 的時間 (毫秒)。如果無法計算該值，則可以是 NaN。	任務、運算子、平行處理層級	* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。 這些指標可用於識別應用程式中的瓶頸。
cpuUtilization	百分比	跨任務管理員的 CPU 使用率整體百分比。例如，如果有 5 個任務管理員，Managed Service for	應用程式	您可以使用此指標來監控應用程式中的最小、平均和最大 CPU 使用率。該CPUUtiliz

指標	單位	描述	Level	使用須知	
		Apache Flink 會針對每個報告間隔發行此指標的 5 個樣本。		`ation` 指標僅考慮在容器內運行的 TaskManager JVM 進程的 CPU 使用率。	

指標	單位	描述	Level	使用須知	
containerCPUUtilization	百分比	Flink 應用程式叢集中跨任務管理員容器的 CPU 使用率整體百分比。例如，如果有五個工作管理員，則相應地有五個 TaskManager 容器，而 Apache Flink 的受管理服務會每 1 分鐘報告間隔發佈 2*5 個此指標的範例。	應用程式	它按每個容器計算如下： 容器使用的總 CPU 時間 (秒) * 100 / 容器 CPU 限制 (單位為 CPU/秒) 該CPUUtilization 指標僅考慮在容器內運行的 TaskManager JVM 進程的 CPU 使用率。同一個容器內的 JVM 外部還有其他元件在執行。containerCPUUtilization 指標為您提供了更完整的視角，包括容器中 CPU 耗盡的所有處理序以及由此引起的故障。	

指標	單位	描述	Level	使用須知
containerMemoryUtilization	百分比	Flink 應用程式叢集中跨任務管理員容器的記憶體使用率整體百分比。例如，如果有五個工作管理員，則相應地有五個 TaskManager 容器，而 Apache Flink 的受管理服務會每 1 分鐘報告間隔發佈 2*5 個此指標的範例。	應用程式	它按每個容器計算如下： 容器記憶體使用量 (位元組) * 100 / 每 Pod 部署規格的容器記憶體限制 (位元組) HeapMemoryUtilization 和 指ManagedMemoryUtilzations 標僅考慮特定的內存指標，例如 TaskManager JVM 的堆內存使用情況或託管內存（如 RockSDB 狀態後端等本機進程的 JVM 外的內存使用情況）。containerMemoryUtilization 指標可為您提供更完整的視角，現包括工作集記憶體，這是一個更佳

指標	單位	描述	Level	使用須知	
				的記憶體總量耗盡追蹤器。一旦耗盡，它將導致該Out of Memory Error網TaskManager繭。	
containerDiskUtilization	百分比	Flink 應用程式叢集中跨任務管理員容器的磁碟使用率整體百分比。例如，如果有五個工作管理員，則相應地有五個TaskManager 容器，而 Apache Flink 的受管理服務會每 1 分鐘報告間隔發佈 2*5 個此指標的範例。	應用程式	它按每個容器計算如下： 磁碟使用量 (位元組) * 100 / 容器的磁碟限制 (位元組) 對於容器，它代表在其上設定容器根磁碟區的檔案系統的使用率。	
currentInputWatermark	毫秒	此應用程式/運算子/任務/執行緒收到的最後一個浮水印	應用程式、運算子、任務、平行處理層級	僅針對具有兩個輸入的維度發出此記錄。這是最後接收到的浮水印的最小值。	

指標	單位	描述	Level	使用須知	
currentOutputWatermark	毫秒	此應用程式/運算子/任務/執行緒發出的最後一個浮水印	應用程式、運算子、工作、平行處理層級		
downtime	毫秒	對於目前處於失敗/復原狀況的作業，此中斷期間經過的時間。	應用程式	此指標衡量作業失敗或復原時經過的時間。此指標針對執行中作業傳回 0，針對完成的作業傳回 -1。如果此指標不是 0 或 -1，則表示應用程式的 Apache Flink 作業執行失敗。	
fullRestarts	計數	此作業提交後完全重新啟動的總次數。此指標不衡量細微的重新啟動。	應用程式	您可以使用此指標評估應用程式總體運作狀態。在 Managed Service for Apache Flink 進行內部維護期間，可能會重新啟動。重新啟動時間高於正常狀態可能表示應用程式發生了問題。	

指標	單位	描述	Level	使用須知	
heapMemoryUtilization	百分比	任務管理員的整體堆積記憶體使用率。例如，如果有 5 個任務管理員，Managed Service for Apache Flink 會針對每個報告間隔發行此指標的 5 個樣本。	應用程式	您可以使用此指標來監控應用程式中的最小、平均和最大堆積記憶體使用率。特定內存指標的HeapMemoryUtilization 唯一帳戶，如 TaskManager JVM 的堆內存使用情況。	
idleTimeMsPerSecond*	毫秒	此任務或運算子每秒閒置 (沒有要處理的資料) 的時間 (毫秒)。閒置時間不包括背壓時間，因此如果任務受到背壓，則不會閒置。	任務、運算子、平行處理層級	* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。 這些指標可用於識別應用程式中的瓶頸。	

指標	單位	描述	Level	使用須知	
lastCheckpointSize	位元組	最後一個檢查點的大小總計	應用程式	您可以使用此指標判斷執行中應用程式的儲存體使用率。 如果此指標的值增加，可能表示應用程式發生了問題，例如記憶體流失或瓶頸。	
lastCheckpointDuration	毫秒	完成最後一個檢查點所花費的時間	應用程式	此指標會測量完成最新檢查點所花費的時間。如果此指標的值增加，這可能表示應用程式發生問題，例如記憶體流失或瓶頸。在某些情況下，您可以藉由停用檢查點來疑難排解此問題。	

指標	單位	描述	Level	使用須知	
managedMemoryUsed*	位元組	目前使用中的受管記憶體數量。	應用程式、運算子、任務、平行處理層級	* 僅適用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。 這與 Java 堆積之外由 Flink 管理的記憶體有關。它用於 RocksDB 狀態後端，並且也可用於應用程式。	

指標	單位	描述	Level	使用須知	
managedMemoryTotal *	位元組	記憶體總量。	應用程式、運算子、工作、平行處理	* 僅適用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。 這與 Java 堆積之外由 Flink 管理的記憶體有關。它用於 RocksDB 狀態後端，也可用於應用程式。ManagedMemoryUtilzations 指標僅考慮特定的記憶體指標，例如受管記憶體 (用於 RocksDB 狀態後端等原生處理序的 JVM 外記憶體使用情況)	

指標	單位	描述	Level	使用須知	
managedMemoryUtilization*	百分比	派生者 managedMemoryUsed/ managedMemoryTotal	應用程式、運算子、任務、平行處理層級	* 僅適用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。 這與 Java 堆積之外由 Flink 管理的記憶體有關。它用於 RocksDB 狀態後端，也可用於應用程式。	
numberOfFailedCheckpoints	計數	檢查點失敗的次數。	應用程式	您可以使用此指標來監控應用程式運作狀態和進度。檢查點可能會因為應用程式問題 (例如輸送量或許可問題) 而失敗。	

指標	單位	描述	Level	使用須知	
numRecordsIn*	計數	此應用程式、運算子或任務已接收的記錄總數。	應用程式、運算子、工作、平行處理	* 若要套用一段時間內 (秒/分鐘) 的 SUM 統計資料： • 選取正確層級的指標。如果要追蹤運算子的指標，則需要選取對應的運算子指標。 • 由於 Managed Service for Apache Flink 每分鐘需要 4 個指標快照，因此應使用下列指標數學表達式：m1/4，其中 m1 是一段期間 (秒/分鐘) 內的 SUM 統計資料 指標的「層級」指定此指標是衡量整個應用程式、特定運算子還是	

指標	單位	描述	Level	使用須知	
				特定任務接收的記錄總數。	

指標	單位	描述	Level	使用須知	
numRecordsInPerSecond*	計數/秒	此應用程式、運算子或任務每秒收到的記錄總數。	應用程式、運算子、工作、平行處理	* 若要套用一段時間內 (秒/分鐘) 的 SUM 統計資料： • 選取正確層級的指標。如果要追蹤運算子的指標，則需要選取對應的運算子指標。 • 由於 Managed Service for Apache Flink 每分鐘需要 4 個指標快照，因此應使用下列指標數學表達式：m1/4，其中 m1 是一段期間 (秒/分鐘) 內的 SUM 統計資料 指標的「層級」指定此指標是衡量整個應用程式、特定運算子還是	

指標	單位	描述	Level	使用須知	
				特定任務每秒接收的記錄總數。	

指標	單位	描述	Level	使用須知	
numRecordsOut*	計數	此應用程式、運算子或任務發出的記錄總數。	應用程式、運算子、工作、平行處理	* 若要套用一段時間內 (秒/分鐘) 的 SUM 統計資料： • 選取正確層級的指標。如果要追蹤運算子的指標，則需要選取對應的運算子指標。 • 由於 Managed Service for Apache Flink 每分鐘需要 4 個指標快照，因此應使用下列指標數學表達式：m1/4，其中 m1 是一段期間 (秒/分鐘) 內的 SUM 統計資料 指標的「層級」指定此指標是衡量整個應用程式、特定運算子還是	

指標	單位	描述	Level	使用須知	
				特定任務發出的記錄總數。	
numLateRecordsDropped*	計數	應用程式、運算子、工作、平行處理		* 若要套用一段時間內 (秒/分鐘) 的 SUM 統計資料： • 選取正確層級的指標。如果要追蹤運算子的指標，則需要選取對應的運算子指標。 • 由於 Managed Service for Apache Flink 每分鐘需要 4 個指標快照，因此應使用下列指標數學表達式：m1/4，其中 m1 是一段期間 (秒/分鐘) 內的 SUM 統計資料 此運算子或任務因遲到而丟棄的記錄數。	

指標	單位	描述	Level	使用須知
numRecordsOutPerSecond*	計數/秒	此應用程式、運算子或任務每秒發出的記錄總數。	應用程式、運算子、工作、平行處理	* 若要套用一段時間內 (秒/分鐘) 的 SUM 統計資料： • 選取正確層級的指標。如果要追蹤運算子的指標，則需要選取對應的運算子指標。 • 由於 Managed Service for Apache Flink 每分鐘需要 4 個指標快照，因此應使用下列指標數學表達式：m1/4，其中 m1 是一段期間 (秒/分鐘) 內的 SUM 統計資料 指標的「層級」指定此指標是衡量整個應用程式、特定運算子還是

指標	單位	描述	Level	使用須知	
				特定任務每秒發出的記錄總數。	
oldGenerationGCCount	計數	所有任務管理員中發生的垃圾回收操作總數。	應用程式		
oldGenerationGCTime	毫秒	執行垃圾回收操作所花費的總時間。	應用程式	您可以使用此指標來監控總計、平均和最大垃圾回收時間。	
threadCount	計數	應用程式使用的即時執行緒總數。	應用程式	此指標衡量應用程式的程式碼使用的執行緒數目。這與應用程式平行處理層級不同。	
uptime	毫秒	作業在不中斷的情況下執行的時間。	應用程式	您可以使用此指標來判斷作業是否在成功執行。此指標針對已完成的作業傳回 -1。	

Kinesis Data Streams 連接器指標

除了下列項目之外，AWS 還會發出 Kinesis Data Streams 的所有記錄：

指標	單位	描述	Level	使用須知
millisbehindLatest	毫秒	取用者位於串流開頭之後的毫秒數，指出取用者落後目前時間多久。	應用程式 (串流)、平行處理 (用於 ShardId)	• 值為 0 表示記錄處理已跟上進度，此時沒有任何新記錄可供處理。可以使用串流名稱和碎片 ID 指定特定碎片的指標。 • 值 -1 表示服務尚未報告指標的值。
bytesRequestedPerFetch	位元組	對 getRecords 的單一呼叫請求的位元組。	應用程式 (串流)、平行處理 (用於 ShardId)	

Amazon MSK 連接器指標

除了以下項目之外，AWS 還會發出 Amazon MSK 的所有記錄：

指標	單位	描述	Level	使用須知
currentoffsets	N/A	每個分割區的取用者目前的讀取位移。您可以依據主題名稱和分割區 ID 來指定特定分割區的指標。	應用程式 (針對主題)、平行處理 (用於 PartitionId)	
commitsFailed	N/A	向 Kafka 遞交位移失敗的總數，	應用程式、運算子、工作、平行處理	將位移遞交回 Kafka 只是公開取用者進度的一

指標	單位	描述	Level	使用須知
		如果啟用了位移遞交和檢查點。		種手段，因此遞交失敗不會影響 Flink 的檢查點分割區位移完整性。
commitsSucceeded	N/A	向 Kafka 成功遞交位移的總數，如果啟用了位移遞交和檢查點。	應用程式、運算子、工作、平行處理	
committedoffsets	N/A	每個分割區最後一次成功提交到 Kafka 的位移。您可以依據主題名稱和分割區 ID 來指定特定分割區的指標。	應用程式 (針對主題)、平行處理 (用於 PartitionId)	
records_lag_max	計數	此視窗中任何分割區以記錄數目而言的最大延遲	應用程式、運算子、工作、平行處理	
bytes_consumed_rate	位元組	每秒使用的主題位元組平均數目	應用程式、運算子、工作、平行處理	

Apache Zeppelin 指標

對於 Studio 筆記本，AWS 會在應用程式層級發出下列指標：KPUs、cpuUtilization、heapMemoryUtilization、oldGenerationGCTime、oldGeneratio和 threadCount。此外，它還會在應用程式層級發出下表中顯示的指標。

指標	單位	描述	Prometheus 名稱
zeppelinCpuUtilization	百分比	Apache Zeppelin 伺服器中 CPU 使用率的整體百分比。	process_cpu_usage
zeppelinHeapMemoryUtilization	百分比	Apache Zeppelin 伺服器的堆積記憶體使用率整體百分比。	jvm_memory_used_bytes
zeppelinThreadCount	計數	Apache Zeppelin 伺服器使用的即時執行緒總數。	jvm_threads_live_threads
zeppelinWaitingJobs	計數	等待執行緒的已排入佇列的 Apache Zeppelin 作業數目。	jetty_threads_jobs
zeppelinServerUptime	秒鐘	伺服器啟動並執行的總時間。	process_uptime_seconds

檢視 CloudWatch 度量

您可以使用 CloudWatch Amazon 主 CloudWatch 控台或AWS CLI.

使用 CloudWatch 主控台檢視指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 指標。
3. 在 Apache Flink 的受管理服務的「按類別分類的CloudWatch 測量結果」窗格中，選擇測量結果類別。
4. 在上方窗格中，向下捲動以檢視完整指標清單。

若要使用 AWS CLI 來檢視指標

- 在命令提示中，使用下列命令。

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

設定 CloudWatch 量度報表層級

您可以控制應用程式建立的應用程式指標層級。Managed Service for Apache Flink 支援下列指標層級：

- 應用程式：應用程式只報告每個應用程式的最高層級指標。依預設，Managed Service for Apache Flink 指標在應用程式層級發佈。
- 任務：應用程式針對使用「任務」指標報告層級定義的指標來報告任務特定的指標維度，例如每秒進出應用程式的記錄數。
- 運算子：應用程式針對以「運算子」指標報告層級定義的指標來報告運算子特定的指標維度，例如每個篩選或對應操作的指標。
- 平行處理層級：應用程式為每個執行緒報告 `Task` 和 `Operator` 層級指標。由於成本過高，平行處理設定超過 64 的應用程式不建議使用此報告層級。

> **Note**
>
> 鑒於服務所產生的指標資料量，您只能使用此指標層級進行疑難排解。您只能使用 CLI 來設定此指標層級。此指標層級在主控台中無法使用。

預設層級為應用程式。應用程式會報告目前層級和所有更高層級的指標。例如，如果報告層級設定為運算子，則應用程式會報告應用程式、任務和運算子指標。

您可以使用動作的`MonitoringConfiguration`參數或`CreateApplication`動作的參數來設定 CloudWatch `MonitoringConfigurationUpdate`量度報告層級。`UpdateApplication`下列`UpdateApplication`動作要求範例會將 CloudWatch 量度報告層級設定為 Task：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "MonitoringConfigurationUpdate": {
            "ConfigurationTypeUpdate": "CUSTOM",
```

```
            "MetricsLevelUpdate": "TASK"
         }
      }
   }
}
```

您也可以使用 CreateApplication 動作的 LogLevel 參數或 UpdateApplication 動作的LogLevelUpdate 參數來設定記錄層級。您可以使用下列日誌層級：

- ERROR：記錄可能復原的錯誤事件。
- WARN：記錄可能導致錯誤的警告事件。
- INFO：記錄資訊事件。
- DEBUG：記錄一般偵錯事件。

如需 Log4j 記錄層級的詳細資訊，請參閱 Apache Log4j 文件中的自訂日誌層級。

在 Amazon Managed Service for Apache Flink 中使用自訂指標

適用於 Apache Flink 的受管理服務會公開 19 個指標 CloudWatch，包括資源使用量和輸送量的度量。此外，您可以建立自己的指標來追蹤應用程式特定的資料，例如處理事件或存取外部資源。

本主題包含下列章節：

- 運作方式
- 範例
- 檢視自訂指標

運作方式

Managed Service for Apache Flink 中的自訂指標使用 Apache Flink 指標系統。Apache Flink 指標具有下列屬性：

- 類型：指標的類型說明衡量和報告資料的方式。可用的 Apache Flink 指標類型包括「計數」、「量計」、「長條圖」和「計量」 如需關於 Apache Flink 指標類型的詳細資訊，請參閱指標類型。

> **Note**
>
> AWS CloudWatch 度量不支援「色階分佈圖」「阿帕奇」Flink 度量類型。 CloudWatch 只能顯示「計數」、「量測計」和「計量」類型的 Apache Flink 度量。

- 範圍：測量結果的範圍包含其 ID 和一組索引鍵值配對，這些索引鍵值配對會指出測量結果的回報方 CloudWatch式。指標的識別碼包含下列項目：
  - 系統範圍，指出報告指標的層級 (例如「運算子」)。
  - 使用者範圍，定義諸如使用者變數或指標群組名稱等屬性。這些屬性使用 `MetricGroup.addGroup(key, value)` 或 `MetricGroup.addGroup(name)` 定義。

  如需指標範圍的詳細資訊，請參閱範圍。

如需關於 Apache Flink 指標的詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的指標。

若要在 Managed Service for Apache Flink 中建立自訂指標，您可以從任何透過呼叫 `GetMetricGroup` 來擴充 `RichFunction` 的使用者函數存取 Apache Flink 指標系統。此方法會傳回可MetricGroup用來建立和註冊自訂量度的物件。適用於 Apache 的受管理服務 Flink 會報告使用群組金鑰`KinesisAnalytics`建立的所有量度。 CloudWatch您定義的自訂指標具有下列特性：

- 您的自訂指標具有指標名稱和群組名稱。這些名稱必須由英數字元組成。
- 您在使用者範圍中定義的屬性 (`KinesisAnalytics`量度群組除外) 會發佈為 CloudWatch 維度。
- 依預設，自訂指標會在`Application`層級發佈。
- 維度 (任務/運算子/平行處理層級) 會根據應用程式的監控層級新增至指標。您可以使用動作的參數或CreateApplication動作的或MonitoringConfiguration參數來設定應用程式的UpdateApplication監視層級。MonitoringConfigurationUpdate

範例

下列程式碼範例示範如何建立可建立並遞增自訂指標的映射類別，以及如何透過將映射類別新增至 `DataStream` 物件，在應用程式中實作映射類別。

記錄計數自訂指標

下列程式碼範例示範如何建立映射類別，以建立可計算資料串流中記錄數目的指標 (功能與 `numRecordsIn` 指標相同)：

```
    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;
        }
    }
```

在上述範例中，valueToExpose 變數會針對應用程式處理的每筆記錄遞增。

定義映射類別之後，您可以建立實作對應的應用程式內串流：

```
DataStream<String> noopMapperFunctionAfterFilter =
    kinesisProcessed.map(new NoOpMapperFunction("FilteredRecords"));
```

如需此應用程式的完整程式碼，請參閱記錄計數自訂指標應用程式。

單字計數自訂指標

下列程式碼範例示範如何建立映射類別，以建立可計算資料串流中字數的指標：

```
private static final class Tokenizer extends RichFlatMapFunction<String, Tuple2<String,
 Integer>> {

            private transient Counter counter;
```

```
            @Override
            public void open(Configuration config) {
                this.counter = getRuntimeContext().getMetricGroup()
                        .addGroup("KinesisAnalytics")
                        .addGroup("Service", "WordCountApplication")
                        .addGroup("Tokenizer")
                        .counter("TotalWords");
            }

            @Override
            public void flatMap(String value, Collector<Tuple2<String, Integer>>out) {
                // normalize and split the line
                String[] tokens = value.toLowerCase().split("\\W+");

                // emit the pairs
                for (String token : tokens) {
                    if (token.length() > 0) {
                        counter.inc();
                        out.collect(new Tuple2<>(token, 1));
                    }
                }
            }
        }
```

在上述範例中，counter 變數會針對應用程式處理的每個單字遞增。

定義映射類別之後，您可以建立實作對應的應用程式內串流：

```
// 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());
```

如需此應用程式的完整程式碼，請參閱單字計數自訂指標應用程式。

檢視自訂指標

應用程式的自訂指標會顯示在AWS/KinesisAnalytics儀表板的「應用程式」 CloudWatch 量度群組下方的「指標」主控台中。

將 CloudWatch 警示與 Amazon 管理服務搭配 Apache Flink 使用

使用 Amazon 指 CloudWatch 標警示，您可以觀看指定期間內的指 CloudWatch 標。警示會根據在數個期間與閾值相關的指標值或表達式值，來執行一個或多個動作。某個動作將通知傳送至 Amazon Simple Notification Service (Amazon SNS) 主題的範例。

如需有關 CloudWatch 警示的詳細資訊，請參閱使用 Amazon CloudWatch 警示。

建議的警示

本節包含用於監控 Managed Service for Apache Flink 應用程式的建議警示。

下表說明了建議的警示，其中包含下列欄位：

- 指標表達式：根據閾值測試的指標或指標表示式。
- 統計值：用來檢查指標的統計值 — 例如平均值。
- 閾值：使用此警示會要求您決定定義預期應用程式效能限制的閾值。您必須在正常情況下監控應用程式，藉此決定此閾值。
- 說明：可能觸發此警示的原因，以及該狀況的可能解決方案。

指標表達式	統計數字	Threshold	描述
`downtime > 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.

指標表達式	統計數字	Threshold	描述
			For troubleshooting, see 應用程式重新啟動.

指標表達式	統計數字	Threshold	描述
`## (numberOf Failed###) > 0`	Average	0	This metric counts the number of failed checkpoints since the application started. Depending on the application, it can be tolerable if checkpoin ts fail occasionally. But if checkpoints are regularly failing, the application is likely unhealthy and needs further attention. We recommend monitorin g RATE(numb erOfFailedCheckpoi nts) to alarm on the gradient and not on absolute values. Recommended for all applications. Use this metric to monitor application health and checkpoin ting progress. The application saves state data to checkpoints when it's healthy. Checkpoin ting can fail due to timeouts if the application isn't making progress in processing the input

指標表達式	統計數字	Threshold	描述
			data. For troublesh ooting, see 檢查點逾時.
`#### numRecord sOutPerSecond` < threshold	Average	The minimum number of records emitted from the applicati on 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 troublesh ooting, see 輸送量太慢.

指標表達式	統計數字	Threshold	描述
`records_lag_max|millisbehindLatest` > threshold	Maximum	The maximum expected latency during normal conditions.	If the application is consuming from Kinesis or Kafka, these metrics indicate if the application is falling behind and needs to be scaled in order to keep up with the current load. This is a good generic metric that is easy to track for all kinds of applications. But it can only be used for reactive scaling, i.e., when the application has already fallen behind. Recommended for all applications. Use the `records_lag_max` metric for a Kafka source, or the `millisbehindLatest` for a Kinesis stream source. Rising above this threshold can indicate that the application isn't making expected progress on the input data. For troubleshooting, see 輸送量太慢.

指標表達式	統計數字	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 too large 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

指標表達式	統計數字	Threshold	描述
			data, or that there are problems with application health such as backpressure. For troubleshooting, see 無限制狀態增長.

指標表達式	統計數字	Threshold	描述
`lastCheckpointSize` > threshold	Maximum	The maximum expected checkpoint size 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 too large 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

指標表達式	統計數字	Threshold	描述
			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 troublesh ooting, see 無限制狀態增長.
`heapMemor yUtilization` > 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 `heapMemor yUtilization` 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 increasin g the application parallelism. For more information about increasing resources, see 擴展.

指標表達式	統計數字	Threshold	描述
`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 擴展.
`threadsCount` > threshold	Maximum	The maximum expected `threadsCount` 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.

指標表達式	統計數字	Threshold	描述
`#oldGarbageCollection## * 100#/60 ###### # '#` > 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.

將自訂訊息寫入 CloudWatch 記錄

您可以將自訂訊息寫入 Apache Flink 應用程式的 CloudWatch 記錄檔的受管理服務。您可以使用 Apache log4j 程式庫或 Simple Logging Facade for Java (SLF4J) 程式庫來執行這項操作。

主題

使用 Log4j 寫入 CloudWatch 日誌

1. 將下列相依項新增至應用程式的 pom.xml 檔案：

```
<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. 包括來自程式庫的物件：

```
import org.apache.logging.log4j.Logger;
```

3. 具現化 Logger 物件，傳入你的應用程式類別：

```
private static final Logger log =
 LogManager.getLogger.getLogger(YourApplicationClass.class);
```

4. 使用 log.info 寫入日誌。大量訊息會寫入應用程式日誌。若要讓您的自訂訊息更易於篩選，請使用INFO應用程式日誌層級。

```
log.info("This message will be written to the application's CloudWatch log");
```

應用程式會將記錄寫入日誌，並顯示類似如下的訊息：

```
{
  "locationInformation": "com.amazonaws.services.managed-
flink.StreamingJob.main(StreamingJob.java:95)",
  "logger": "com.amazonaws.services.managed-flink.StreamingJob",
  "message": "This message will be written to the application's CloudWatch log",
  "threadName": "Flink-DispatcherRestEndpoint-thread-2",
  "applicationARN": "arn:aws:kinesisanalyticsus-east-1:123456789012:application/test",
  "applicationVersionId": "1", "messageSchemaVersion": "1",
  "messageType": "INFO"
}
```

使用 SLF4J 寫入 CloudWatch 記錄檔

1. 將下列相依項新增至應用程式的 pom.xml 檔案：

   ```
   <dependency>
       <groupId>org.slf4j</groupId>
       <artifactId>slf4j-log4j12</artifactId>
       <version>1.7.7</version>
       <scope>runtime</scope>
   </dependency>
   ```

2. 包括來自程式庫的物件：

   ```
   import org.slf4j.Logger;
   import org.slf4j.LoggerFactory;
   ```

3. 具現化 Logger 物件，傳入你的應用程式類別：

   ```
   private static final Logger log =
    LoggerFactory.getLogger(YourApplicationClass.class);
   ```

4. 使用 log.info 寫入日誌。大量訊息會寫入應用程式日誌。若要讓您的自訂訊息更易於篩選，請使用INFO應用程式日誌層級。

   ```
   log.info("This message will be written to the application's CloudWatch log");
   ```

應用程式會將記錄寫入日誌，並顯示類似如下的訊息：

```
{
  "locationInformation": "com.amazonaws.services.managed-
flink.StreamingJob.main(StreamingJob.java:95)",
  "logger": "com.amazonaws.services.managed-flink.StreamingJob",
  "message": "This message will be written to the application's CloudWatch log",
  "threadName": "Flink-DispatcherRestEndpoint-thread-2",
  "applicationARN": "arn:aws:kinesisanalyticsus-east-1:123456789012:application/test",
  "applicationVersionId": "1", "messageSchemaVersion": "1",
  "messageType": "INFO"
}
```

使用 AWS CloudTrail 記錄 Managed Service for Apache Flink API 呼叫

Apache Flink 的受管理服務整合在一起AWS CloudTrail，這項服務可提供 Apache Flink 受管理服務中使用者、角色或AWS服務所採取的動作記錄。 CloudTrail 會將 Apache Flink 受管理服務的所有 API 呼叫擷取為事件。擷取的呼叫包括來自 Managed Service for Apache Flink 主控台的呼叫，以及對 Managed Service for Apache Flink API 操作的程式碼呼叫。如果您建立追蹤，您可以啟用持續向 Amazon S3 儲存貯體傳遞事件，包括 Apache Flink 受管服務的事件。 CloudTrail 如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。使用收集的資訊 CloudTrail，您可以判斷對 Apache Flink 的受管理服務提出的要求、提出要求的 IP 位址、提出要求的人員、提出要求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail用者指南。

中的 Apache 快速連結資訊的管理服務 CloudTrail

CloudTrail 在您創建AWS帳戶時，您的帳戶已啟用。當活動在 Apache Flink 的受管理服務中發生時，該活動會與事件歷史記錄中的其他AWS服務 CloudTrail 事件一起記錄在事件中。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱檢視具有事 CloudTrail 件記錄的事件。

如需 AWS 帳戶中正在進行事件的記錄 (包括 Managed Service for Apache Flink 的事件)，請建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。根據預設，當您在主控台建立線索時，線索會套用到所有 AWS 區域。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他AWS服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合

- 設定的 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 記錄檔並從多個帳戶接收 CloudTrail 記錄檔

所有適用於 Apache Flink 的受管理服務動作都會記錄下來， CloudTrail 並記錄在適用於 Apache Flink API 的受管理服務參考資料中。例如，呼叫CreateApplication和動 UpdateApplication作會在 CloudTrail 記錄檔中產生項目。

每一筆事件或日誌項目都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否透過根或 AWS Identity and Access Management (IAM) 使用者憑證來提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務提出。

如需詳細資訊，請參閱 CloudTrail 使用者身分元素。

了解 Managed Service for Apache Flink 日誌檔案項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示示範AddApplicationCloudWatchLoggingOption和DescribeApplication動作的 CloudTrail 記錄項目。

```
{
    "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": "kinesisanlaytics.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:kinesisanalyticsus-
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": "kinesisanlaytics.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"
        }
    ]
}
```

調整 Amazon Managed Service for Apache Flink 中的效能

本主題說明可監控和改善 Managed Service for Apache Flink 應用程式效能的技術。

主題

效能疑難排解

本節包含可以透過檢查來診斷和修正效能問題的徵狀清單。

如果資料來源是 Kinesis 串流，效能問題通常呈現為 `millisbehindLatest` 指標高或增加。對於其他來源，您可以檢查代表從源讀取時滯後的類似指標。

資料路徑

調查應用程式的效能問題時，請考慮資料所採用的整個路徑。下列應用程式元件如果未正確設計或佈建，可能會成為效能瓶頸並造成背壓：

- 資料來源和目的地：確保與應用程式互動的外部資源是針對應用程式將遇到的輸送量而正確佈建。
- 狀態資料：確保應用程式不會太頻繁地與狀態存放區互動。

  您可以最佳化應用程式正在使用的序列化程式。預設的 Kryo 序列化程式可以處理任何可序列化類型，但是如果應用程式只將資料存儲在 POJO 類型中，則可以使用更高性能的序列化程式。如需關於 Apache Flink 序列化程式的詳細資訊，請參閱《Apache Flink 文件》中的資料類型與序列化。https://ci.apache.org/projects/flink/flink-docs-release-1.8/
- 運算子：確保運算子實作的業務邏輯不是太複雜，或者您不會在處理每筆記錄時建立或使用資源。還要確保應用程式不會太頻繁地建立滑動或輪轉視窗。

效能疑難排解解決方案

本節包含效能問題的潛在解決方案。

主題

- CloudWatch 監控層級
- 應用程式 CPU 指標
- 應用程式平行處理層級
- 應用程式記錄
- 運算子平行處理層級
- 應用程式邏輯
- 應用程式記憶體

CloudWatch 監控層級

確認未將 CloudWatch 監控層級設定得太詳細。

Debug 監控日誌層級設定會產生大量的流量，這會造成背壓。您只能在主動調查應用程式問題時使用它。

如果您的應用程式具有較高的 Parallelism 設定值，使用 Parallelism 監控指標層級也會產生大量流量，進而導致背壓。只有在您的應用程式 Parallelism 不足或調查應用程式問題時，才使用此指標層級。

如需詳細資訊，請參閱應用程式監控層級。

應用程式 CPU 指標

檢查應用程式的 CPU 指標。如果此指標高於 75%，您可以啟用自動擴展，允許應用程式為自己配置更多資源。

如果啟用了自動擴展，則當 CPU 使用率超過 75% 並持續 15 分鐘時，應用程式會配置更多資源。如需關於擴展的詳細資訊，請參閱下面的適當管理擴展一節和擴展。

> **Note**
>
> 應用程式只會根據 CPU 使用率自動擴展。應用程式不會自動擴展以回應其他系統指標，例如 heapMemoryUtilization。如果應用程式在其他指標上具有較高的使用量，請手動增加應用程式的平行處理層級。

應用程式平行處理層級

增加應用程式的平行處理層級。您可以使用 UpdateApplication 動作的 `ParallelismConfigurationUpdate` 參數來更新應用程式的平行處理層級。

應用程式的最大 KPU 預設為 64 個，可透過請求提高限制來增加。

務必基於每個運算子的工作負載指派運算子的平行處理層級，而不僅僅是單獨增加應用程式平行處理層級。請參閱下文的運算子平行處理層級。

應用程式記錄

檢查應用程式是否正在記錄正在處理的每筆記錄項目。在應用程式具有高輸送量時，為每筆記錄寫入日誌項目將會導致資料處理中出現嚴重瓶頸。若要檢查此狀況，請查詢您的日誌，找出應用程式隨處理的每筆記錄所寫入的日誌項目。如需關於讀取應用程式日誌的詳細資訊，請參閱the section called “分析日誌”。

運算子平行處理層級

確認應用程式的工作負載在工作者處理序之間平均分配。

如需調整應用程式運算子工作負載的相關資訊，請參閱運算子擴展。

應用程式邏輯

檢查您的應用程式邏輯是否有效率低或性能不佳的操作，例如存取外部相依項（例如資料庫或 Web 服務），存取應用程式狀態等。如果外部相依項效能不高或無法可靠地存取，也可能會阻礙效能，這可能導致外部相依項傳回 `HTTP 500` 錯誤。

如果您的應用程式使用外部相依項來富集或以其他方式處理傳入資料，請考慮改用非同步 IO。如需詳細資訊，請參閱《Apache Flink 文件》中的非同步 I/Ohttps://ci.apache.org/projects/flink/flink-docs-release-1.8/。

應用程式記憶體

檢查應用程式是否有資源洩漏。如果應用程式未正確處置執行緒或記憶體，您可能會看到 `millisbehindLatest`、`CheckpointSize` 和`CheckpointDuration`指標急遽增加或逐漸增加。此情況還可能導致任務管理員或作業管理員失敗。

效能最佳實務

本節說明針對效能設計應用程式時的特殊考量。

適當管理擴展

本節包含管理應用程式層級和運算子層級擴展的相關資訊。

本節包含下列主題：

- 適當管理應用程式擴展
- 適當管理運算子擴展

適當管理應用程式擴展

您可以使用自動擴展來處理應用程式活動中的意外尖峰。如果符合下列條件，應用程式的 KPU 會自動增加：

- 已為應用程式啟用自動擴展。
- CPU 使用率在 15 分鐘內保持在 75% 以上。

如果已啟用自動擴展，但 CPU 使用率未維持在此閾值，則應用程式將不會縱向擴展 KPU。如果您遇到不符合此閾值的 CPU 使用率尖峰，或是不同使用量指標 (例如 `heapMemoryUtilization`) 中出現尖峰的情況，請手動增加擴展以允許應用程式處理活動尖峰。

> **Note**
>
> 如果應用程式透過自動擴展自動新增了更多資源，則會在閒置一段時間後釋出新資源。縮減資源會暫時影響效能。

如需擴展的詳細資訊，請參閱擴展。

適當管理運算子擴展

您可以透過驗證應用程式的工作負載在工作者處理序之間平均分配，以及應用程式中的運算子擁有穩定且高效能狀態所需的系統資源，藉此改善應用程式的效能。

您可以使用 `parallelism` 設定為應用程式程式碼中的每個運算子設定平行處理層級。如果您未為運算子設定平行處理層級，它就會使用應用程式層級的平行處理設定。使用應用程式層級平行處理設定的運算子可能會使用應用程式可用的所有系統資源，這會導致應用程式不穩定。

為了準確確定每個運算子的平行處理層級，請考慮該運算子與應用程式中其他運算子相比的相對資源需求。對比耗費更少資源的運算子，為耗費更多資源的運算子設定更高的運算子平行處理設定。

應用程式的運算子平行處理層級總數是應用程式中所有運算子的平行處理層級之總和。您可以決定應用程式的運算子平行處理層級總數與應用程式可用的任務空位總數之間的最佳比率，以調整應用程式的運算子平行處理層級總數。運算子平行處理層級與任務空位的典型穩定比率為 4:1，也就是說，應用程式每四個可用的運算子子任務就有一個任務空位可用。具有耗費更多資源的運算子的應用程式可能需要 3:1 或 2:1 的比率，而具有耗費更少資源的運算子的應用程式在比率為 10:1 時可能是穩定的。

您可以使用 執行時間屬性 設定運算子的比率，以便調整運算子的平行處理層級，而無需編譯和上傳應用程式程式碼。

下列程式碼範例示範如何將運算子平行處理層級設定為目前應用程式平行處理層級的可調整比率：

```
Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
operatorParallelism =
    StreamExecutionEnvironment.getParallelism() /
    Integer.getInteger(

 applicationProperties.get("OperatorProperties").getProperty("MyOperatorParallelismRatio")
    );
```

如需關於子任務、任務空位和其他應用程式資源的資訊，請參閱應用程式資源。

若要控制整個應用程式工作者處理序的工作負載分配，請使用 `Parallelism` 設定和 `KeyBy` 分割方法。如需詳細資訊，請參閱《Apache Flink 文件》中的下列主題https://ci.apache.org/projects/flink/flink-docs-release-1.8/：

- 平行執行
- DataStream 轉換

監控外部相依項資源使用量

如果目的地 (例如 Kinesis 串流、Kinesis Data Firehose、DynamoDB 或 OpenSearch 服務) 存在效能瓶頸，應用程式將會遇到背壓。確認已針對應用程式輸送量正確佈建您的外部相依項。

> **Note**
>
> 其他服務中的故障可能會導致應用程式失敗。如果您在應用程式中看到故障，請檢查目的地服務的 CloudWatch 日誌查看故障。

在本機執行 Apache Flink 應用程式

若要疑難排解記憶體問題，您可以在本機 Flink 安裝中執行應用程式。這可讓您存取堆疊追蹤和堆積傾印等偵錯工具，在 Managed Service for Apache Flink 中執行應用程式時,這些工具不可用。

如需關於建立本機 Flink 安裝的資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的本機設定教學課程。

監控效能

本節說明監控應用程式效能的工具。

使用 CloudWatch 指標監控效能

您可以使用 CloudWatch 指標監控應用程式的資源使用量、輸送量、檢查點和停機時間。如需將 CloudWatch 指標與 Managed Service for Apache Flink 應用程式搭配使用的相關資訊，請參閱Managed Service for Apache Flink 中的指標和維度。

使用 CloudWatch 日誌和警示監控效能

您可以使用 CloudWatch Logs 監控可能導致效能問題的錯誤情況。

當 Apache Flink 作業狀態從 `RUNNING` 狀態變更為 `FAILED` 狀態時，日誌項目中會顯示錯誤情況。

您可以使用 CloudWatch 警示來建立效能問題的通知，例如資源使用或檢查點指標超出安全閾值，或未預期的應用程式狀態變更。

如需為 Managed Service for Apache Flink 應用程式建立 CloudWatch 警示的相關資訊，請參閱警示。

Managed Service for Apache Flink 和 Studio 筆記本配額

使用 Amazon Managed Service for Apache Flink 時，請注意下列配額：

- 您可以為帳戶中的每個區域建立最多 50 個 Managed Service for Apache Flink 應用程式。您可以透過服務配額提高表單來建立案例，以請求額外的應用程式。如需詳細資訊，請參閱 AWS Support 中心。

  如需支援 Managed Service for Apache Flink 的區域清單，請參閱 Managed Service for Apache Flink 區域和端點。

- 依預設，Kinesis 處理單元 (KPU) 的數目限制為 64。如需如何請求提高此配額的指示，請參閱 Service Quotas 中的請求提高配額。請確定您已指定需要套用新 KPU 限制的應用程式前綴。

  使用 Managed Service for Apache Flink，您的 AWS 帳戶需要為配置的資源 (而不是應用程式使用的資源) 支付費用。我們會根據您執行串流處理應用程式所使用的 KPU 數目上限，以小時費率計費。單一 KPU 可為您提供 1 個 vCPU 和 4 GiB 的記憶體。此服務也會針對每個 KPU 提供 50 GiB 的執行中應用程式儲存體。

- 您可以為每個應用程式建立多達 1,000 個 Managed Service for Apache Flink 快照。

- 您可以為每個應用程式指派最多 50 個標籤。

- 應用程式 JAR 檔案的大小上限為 512 MiB。如果超出此配額，應用程式將無法啟動。

對於 Studio 筆記本，適用下列配額。若要請求提高配額，請建立支援案例。

- `websocketMessageSize` = 5 MiB
- `noteSize` = 5 MiB
- `noteCount` = 1000

- `Max cumulative UDF size` = 100 MiB
- `Max cumulative dependency jar size` = 300 MiB

Managed Service for Apache Flink 維護

Managed Service for Apache Flink 會定期使用作業系統和容器映像安全性更新來修補您的應用程式，以維持合規性並達到 AWS 安全目標。下表列出 Managed Service for Apache Flink 執行此類維護的預設時間範圍。應用程式維護可能會在與您的地區對應的時間範圍內隨時發生。在此維護程序期間，應用程式可能會遇到 10 到 30 秒的停機時間。但是，實際停機時間取決於應用程式狀態。如需如何將此停機時間所造成的影響降到最低的資訊，請參閱the section called “容錯能力：檢查點和儲存點”。

若要變更 Managed Service for Apache Flink 對您的應用程式執行維護的時間範圍，請使用 UpdateApplicationMaintenanceConfiguration API。

區域	維護時間範圍
AWS GovCloud (美國西部)	上午 6 時至下午 2 時 (UTC)
AWS GovCloud (美國東部)	上午 3 時至 11 時 (UTC)
美國東部 (維吉尼亞北部)	上午 3 時至 11 時 (UTC)
美國東部 (俄亥俄)	上午 3 時至 11 時 (UTC)
美國西部 (加利佛尼亞北部)	上午 6 時至下午 2 時 (UTC)
美國西部 (奧勒岡)	上午 6 時至下午 2 時 (UTC)
亞太區域 (香港)	下午 1 時至 9 時 (UTC)
亞太區域 (孟買)	下午 4 時 30 分至上午 12 時 30 分 (UTC)
亞太區域 (海德拉巴)	下午 4 時 30 分至上午 12 時 30 分 (UTC)
亞太區域 (首爾)	下午 1 時至 9 時 (UTC)
亞太區域 (新加坡)	下午 2 時至 10 時 (UTC)
亞太區域 (雪梨)	中午 12 時至下午 8 時 (UTC)
亞太區域 (雅加達)	下午 3 時至晚上 11 時 (UTC)
亞太區域 (東京)	下午 1 時至 9 時 (UTC)

區域	維護時間範圍
加拿大 (中部)	上午 3 時至 11 時 (UTC)
中國 (北京)	下午 1 時至 9 時 (UTC)
中國 (寧夏)	下午 1 時至 9 時 (UTC)
歐洲 (法蘭克福)	上午 6 時至下午 2 時 (UTC)
歐洲 (蘇黎世)	下午 8 時至次日凌晨 4 時 (UTC)
歐洲 (愛爾蘭)	下午 10 時至次日上午 6 時 (UTC)
歐洲 (倫敦)	下午 10 時至次日上午 6 時 (UTC)
歐洲 (斯德哥爾摩)	下午 11 時至次日上午 7 時 (UTC)
歐洲 (米蘭)	下午 9 時至上午 5 時 (UTC)
歐洲 (西班牙)	下午 9 時至上午 5 時 (UTC)
非洲 (開普敦)	下午 8 時至次日凌晨 4 時 (UTC)
歐洲 (愛爾蘭)	下午 10 時至次日上午 6 時 (UTC)
歐洲 (倫敦)	下午 11 時至次日上午 7 時 (UTC)
歐洲 (巴黎)	下午 11 時至次日上午 7 時 (UTC)
歐洲 (斯德哥爾摩)	下午 11 時至次日上午 7 時 (UTC)
中東 (巴林)	下午 1 時至 9 時 (UTC)
中東 (阿拉伯聯合大公國)	下午 6 時至次日凌晨 2 時 (UTC)
南美洲 (聖保羅)	下午 7 時至次日凌晨 3 時 (UTC)
以色列 (特拉維夫)	下午 8 時至次日凌晨 4 時 (UTC)

為所有運算子設定 UUID

當 Managed Service for Apache Flink 為具有快照的應用程式啟動 Flink 作業時，Flink 作業可能會因為某些問題而無法啟動。其中一個問題是運算符 ID 不符。Flink 預期 Flink 作業圖表運算子具有明確且一致的運算子 ID。如果未明確設定，Flink 會自動產生運算子 ID。這是因為，Flink 使用這些運算子 ID 來唯一識別作業圖表中的運算子，並使用它們將每個運算子的狀態儲存在儲存點中。

當 Flink 在作業圖表的運算子 ID 與儲存點中定義的運算子 ID 之間找不到 1:1 對應時，就會發生運算子 ID 不符問題。當未設定明確一致的運算子 ID，且 Flink 自動產生的運算子 ID 可能與每個作業圖標建立的 ID 不一致時，就會發生這種情況。在維護執行時間間，應用程式遇到此問題的可能性很高。為了避免這種情況，我們建議客戶在 flink 程式碼中為所有運算子設定 UUID。如需詳細資訊，請參閱生產就緒性下的為所有運算子設定 UUID 主題。

生產就緒性

這是在 Managed Service for Apache Flink 上執行生產應用程式的重要方面的集合。這不是一份詳盡清單，而是在將應用程式投入生產環境之前應該注意的最基本的事項。

負載測試應用程式

應用程式的某些問題僅在高負載下才會表現出來。我們已經看到應用程式看起來運作狀態良好的情況，而某個操作事件顯著放大了應用程式的負載。這種情況的發生可能與應用程式本身完全無關：如果資料來源或資料接收器在幾個小時內無法使用，Flink 應用程式將無法進展。一旦該問題得到修正，大量累積的未處理資料待處理，這可能會完全耗盡可用的資源。然後，該負載就會放大以前沒有出現的錯誤或效能問題。

因此，對生產應用程式執行適當的負載測試至關重要。在這些負載測試期間應回答的問題包括：

- 應用程式在持續高負載下是否穩定？
- 應用程式在尖峰負載下是否仍可取得儲存點？
- 處理 1 小時的待辦項目需要多長時間？ 如果是 24 小時，將需要多長時間 (取決於串流中資料的最大保留時間)？
- 應用程式擴展時，其輸送量是否會增加？

從資料串流使用時，可以透過產生到串流中一段時間來模擬這些案例。然後啟動應用程式並讓它從時間開始時消耗資料，例如在 Kinesis 資料串流的情況下，使用 `TRIM_HORIZON` 的開始位置。

最大平行處理層級

最大平行處理層級定義具有狀態的應用程式可擴展至的最大平行處理層級。這在首次建立狀態時定義，並且無法在不放棄狀態的情況下將運算子擴展到超出此最大值。

最大平行處理層級在首次建立狀態時設定。

依預設，「最大平行處理層級」設定為：

- 128：如果平行處理層級 <= 128
- `MIN(nextPowerOfTwo(parallelism + (parallelism / 2)), 2^15)`：如果平行處理層級 > 128

如果您打算將應用程式平行處理層級擴展到 > 128，則應明確定義「最大平行處理層級」。

「最大平行處理層級」可以使用 `env.setMaxParallelism(x)` 或單個運算子在應用程式層級定義。除非另有指定，否則所有運算子都會繼承應用程式的最大平行處理層級。

如需詳細資訊，請參閱 Flink 文件中的設定明確的最大平行處理層級。

為所有運算子設定 UUID

在 Flink 將儲存點映射回個別運算子的操作中，會使用 UUID。為每個運算子設定特定的 UUID 可以為要還原的儲存點程序提供穩定映射。

```
.map(...).uid("my-map-function")
```

如需詳細資訊，請參閱生產就緒性檢查清單。

Managed Service for Apache Flink 最佳實務

本節包含開發穩定、高效能的 Managed Service for Apache Flink 應用程式的相關資訊和建議。

主題

容錯能力：檢查點和儲存點

使用檢查點和儲存點在 Managed Service for Apache Flink 應用程式中實作容錯能力。開發和維護應用程式時，請謹記下列各項：

- 建議始終為應用程式啟用檢查點。檢查點可在排程的維護期間以及因服務問題、應用程式相依性失敗及其他問題而發生意外失敗的情況下，為應用程式提供容錯能力。如需排程維護的詳細資訊，請參閱維護。
- 設置 ApplicationSnapshotConfiguration：：`false`在應SnapshotsEnabled用程序開發或故障排除期間。每次應用程式停止時都會建立快照，如果應用程式處於運作狀態不佳或效能不佳，這可能會造成問題。當應用程式進入生產環境且狀態穩定之後，將 `SnapshotsEnabled` 設定為 `true`。

> **Note**
>
> 建議您的應用程式每天建立數次快照，以便使用正確的狀態資料正確重新啟動。快照的正確頻率取決於應用程式的業務邏輯。頻繁拍攝快照可讓您復原較新的資料，但會增加成本並需要更多的系統資源。

如需監控應用程式停機時間的資訊，請參閱 Managed Service for Apache Flink 中的指標和維度

如需實作容錯能力的詳細資訊，請參閱 容錯能力。

不受支援的連接器版本

如果應用程式使用綁定在應用程式 JAR 中的不支援 Kinesis 連接器版本，Managed Service for Apache Flink 1.15 版本將會自動阻止應用程式啟動或更新。升級至 Managed Service for Apache Flink 1.15 版時，確保使用的是最新的 Kinesis 連接器。這是指 1.15.2 版本或更新版本。Managed Service for Apache Flink 將不支援所有其他版本，因為這些版本可能會造成一致性問題或失敗 (使用儲存點停止功能會阻止乾淨停止/更新操作)。

效能與平行處理層級

應用程式可透過調整其平行處理層級並避免效能缺陷來進行擴展，以滿足任何輸送量水平。開發和維護應用程式時，請謹記下列各項：

- 確認您的所有應用程式來源和接收器都已充分佈建且未受到限流。如果來源和接收器是其他AWS服務，請使用 CloudWatch.
- 對於具有非常高的平行處理層級的應用程式，請檢查該高平行處理層級是否已套用到應用程式中的所有運算子。根據預設，Apache Flink 會對應用程式圖形中的所有運算子套用相同的應用程式平行處理層級。這可能會導致來源或接收器的佈建問題，或導致運算子資料處理出現瓶頸。您可以使用 setParallelism 來變更程式碼中每個運算子的平行處理層級設定。
- 了解應用程式中運算子平行處理層級設定的意義。如果您變更運算子的平行處理層級，則當運算子的平行處理層級與目前設定不相容時，可能無法從建立的快照還原應用程式。如需關於設定運算子平行處理層級的詳細資訊，請參閱明確設定運算子的最大平行處理層級。

如需實作擴展的詳細資訊，請參閱 擴展。

設定每個運算子的平行處理層級

根據預設，所有運算子都會在應用程式層級設定平行處理層級。您可以使用 DataStream API 覆寫單一運算子的平行處理原則。`.setParallelism(x)`您可以將運算子平行處理層級設定為等於或低於應用程式平行處理層級的任何平行處理層級。

如果有可能，請將運算子平行處理層級定義為應用程式平行處理層級的函數。如此一來，運算子平行處理層級會隨應用程式平行處理層級而改變。例如，如果您使用自動擴展，則所有運算子都會以相同比例變更其平行處理層級：

```
int appParallelism = env.getParallelism();
...
...ops.setParalleism(appParallelism/2);
```

在某些情況下，您可能需要將運算子並行處理原則設定為常數。例如，將 Kinesis 串流來源的平行處理層級設定為碎片數目。在這些情況下，您應該考慮將運算子平行處理層級作為應用程式設定參數傳遞，以便在不更改程式碼的情況下變更該層級，例如對來源串流重新碎片 (如有需要)。

日誌

您可以使用 CloudWatch 記錄監視應用程式的效能和錯誤狀況。為應用程式設定記錄時，請謹記下列各項：

- 啟用應用程式的 CloudWatch 記錄功能，以便偵錯任何執行階段問題。
- 請勿為應用程式中正在處理的每筆記錄建立日誌項目。這會在處理期間造成嚴重的瓶頸，可能導致處理資料時產生背壓。
- 創建 CloudWatch 警報以在應用程序未正常運行時通知您。如需更多資訊，請參閱 警示

如需實作記錄的詳細資訊，請參閱 記錄和監控。

編碼

您可以使用建議的程式設計做法，讓應用程式具備高效能和穩定性。撰寫應用程式的程式碼時，請謹記以下事項：

- 請勿在應用程式的程式碼、應用程式的 `main` 方法或使用者定義的函數中使用 `system.exit()`。如果想要從程式碼中關閉應用程式，請擲回衍生自 `Exception` 或 `RuntimeException` 的例外狀況，在其中包含關於應用程式所發生問題的訊息。

請注意下列有關服務如何處理此例外狀況的事項：

- 如果從應用程式的 `main` 方法擲回例外狀況，服務會在應用程式轉換至 `RUNNING` 狀態時將其包裝在一個 `ProgramInvocationException` 中，並且作業管理員將無法提交作業。
- 如果從使用者定義的函數擲回例外狀況，作業管理員會讓作業失敗然後重新啟動它，並將例外狀況的詳細資訊寫入例外狀況日誌中。

- 考慮遮蔽您的應用程式 JAR 檔案及其包含的相依性。如果應用程式與 Apache Flink 執行時間之間的套件名稱有可能發生衝突，建議使用遮蔽。如果發生衝突，您的應用程式日誌可能包含 `java.util.concurrent.ExecutionException` 類型的例外狀況。如需遮蔽應用程式 JAR 檔案的詳細資訊，請參閱 Apache Maven Shade 外掛程式。

管理憑證

您不應將任何長期憑證封裝到生產 (或任何其他) 應用程式中。長期憑證可能簽入版本控制系統，很容易丟失。反之，您可以將角色與 Managed Service for Apache Flink 應用程式建立關聯，並為該角色授與權限。然後，執行中的 Flink 應用程式可以從環境中取得具有相應許可的臨時憑證。如果未與 IAM 原生整合的服務需要驗證，例如需要使用者名稱和密碼進行身分驗證的資料庫，您應該考慮將機密儲存在 AWSSecrets Manager 中。

許多 AWS 原生服務都支援驗證：

- Kinesis Data Streams — .java ProcessTaxiStream
- Amazon MSK — https://github.com/aws/aws-msk-iam-auth/using-the-amazon-msk#-library-for-iam-authentication
- Amazon Elasticsearch Service-.java AmazonElasticsearchSink
- Amazon S3：在 Managed Service for Apache Flink 上立即可用

從具有較少碎片/分割區的來源讀取

從 Apache Kafka 或 Kinesis 資料串流讀取時，串流的平行處理層級 (也就是 Kafka 的分割區數目和 Kinesis 的碎片數目) 與應用程式的平行處理層級之間可能不相符。使用單純的設計，應用程式的平行處理層級無法擴展到串流的平行處理層級：來源運算子的每個子任務只能從 1 個或多個碎片/分割區中讀取。這意味著對於只有 2 個碎片的串流和一個平行處理層級為 8 的應用程式，只有兩個子任務實際上從串流中取用資料，有 6 個子任務保持閒置狀態。這可能會大幅限制應用程式的輸送量，特別是如果還原序列化昂貴且由來源執行 (這是預設情況) 時。

為了減輕這種影響，您可以擴展串流。但是，這可能並不總是期望的或可行的。或者，您可以重新構建來源，以便它不執行任何序列化，只是傳遞 byte[]。然後，您可以重新平衡資料，將資料平均分配到所有任務，然後還原序列化該處的資料。透過這種方式，您可以利用所有子任務進行還原序列化，並且這種潛在代價高昂的操作不再受串流的碎片/分割區數量的限制。

Studio 筆記本重新整理間隔

如果要變更段落結的果重新整理間隔，請將其設定為至少 1000 毫秒的值。

Studio 筆記本最佳效能

我們使用以下陳述式進行了測試，並在 events-per-second 乘以 number-of-keys 的結果低於 25,000,000 時取得了最佳效能。這是針對 events-per-second 低於 150,000 以下的情況。

```
SELECT key, sum(value) FROM key-values GROUP BY key
```

浮水印策略和閒置碎片如何影響時間範圍

從 Apache Kafka 和 Kinesis Data Streams 讀取事件時，來源可以根據串流的屬性設定事件時間。在 Kinesis 的情況下，事件時間等於事件的大約到達時間。但是，在來源處設定事件的事件時間無法讓 Flink 應用程式使用事件時間。來源還必須產生浮水印，將事件時間的資訊從來源傳播到所有其他運算子。Flink 文件中對該過程的工作原理進行了清楚的概述。

根據預設，從 Kinesis 讀取的事件時間戳記會設定為 Kinesis 決定的大約到達時間。在應用程式中使用事件時間的其他先決條件是浮水印策略。

```
WatermarkStrategy<String> s = WatermarkStrategy
    .<String>forMonotonousTimestamps()
    .withIdleness(Duration.ofSeconds(...));
```

浮水印策略然後會套用至具有 assignTimestampsAndWatermarks 方法的 DataStream。目前提供了一些有用的內建的策略：

- forMonotonousTimestamps() 將只使用事件時間 (大約到達時間)，並定期發出最大值作為浮水印 (針對每個特定的子任務)
- forBoundedOutOfOrderness(Duration.ofSeconds(...)) 與之前的策略類似，但是將使用事件時間，即產生浮水印的持續時間。

這種方法可行，但有幾個警告需要注意。浮水印在子任務層級產生，並流經運算子圖表。

從 Flink 文件中：

來源函數的每個平行子任務通常會獨立生成其浮水印。這些浮水印會定義該特定平行來源的事件時間。

當浮水印流經串流傳輸程序時，它們會將所到達之運算子處的事件時間提前。每當運算子提前其事件時間時，都會為其後續運算子產生新的浮水印。

某些運算子會取用多個輸入串流；例如聯集，或跟隨 keyBy (...) 或 partition (...) 函數的運算子。這類運算子的目前事件時間是其輸入串流事件時間的最小值。隨著其輸入串流更新其事件時間，運算子的事件時間也會更新。

這意味著，如果來源子任務從閒置碎片中取用，則下游運算子不會從該子任務中收到新的浮水印，因此對使用時間範圍的所有下游運算子進行的處理將停止。為了避免這種情況，客戶可以將 `withIdleness` 選項新增到浮水印策略中。使用該選項，運算子在計算其事件時間時會將浮水印從閒置的上游子任務中排除。空閒子任務不再阻止下游運算子事件時間的提前。

但是，如果沒有子任務正在讀取任何事件 (即串流中沒有事件)，則內建水印策略的閒置選項不會提前事件時間。對於從串流中讀取一組有限事件的測試用例，這一點變得特別明顯。由於最後一個事件讀取後，事件時間不會提前，因此最後一個視窗 (包含最後一個事件) 永遠不會關閉。

Summary

- 如果碎片處於空閒狀態，`withIdleness` 設定將不會產生新的浮水印，它只會從下游運算子的最小浮水印計算中排除閒置子任務發送的最後一個浮水印
- 使用內建的浮水印策略，最後一個開啟的視窗將永遠不會關閉 (除非將發送會將浮水印時間提前的新事件，但會建立一個新視窗，然後保持開啟狀態)
- 即使時間是由 Kinesis 串流設定，如果一個碎片的取用速度比其他碎片快 (例如，在應用程式初始化期間或使用 `TRIM_HORIZON`，其中所有現有碎片被平行取用而忽略其父/子關係的情況下)，則延遲到達事件仍然可能發生
- 浮水印策略的 `withIdleness` 設定似乎取代了閒置碎片的 Kinesis 來源特定的設定 (`ConsumerConfigConstants.SHARD_IDLE_INTERVAL_MILLIS`

範例

下列應用程式正在從串流讀取，並根據事件時間建立工作階段視窗。

```
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);
            }
        }
    });
```

在下列範例中，8 個事件會寫入一個有 16 個碎片的串流 (開頭 2 個和最後一個事件發生在相同的碎片中)。

```
$ aws kinesis put-record --stream-name hp-16 --partition-key 1 --data MQ==
```

```
$ aws kinesis put-record --stream-name hp-16 --partition-key 2 --data Mg==
$ aws kinesis put-record --stream-name hp-16 --partition-key 3 --data Mw==
$ date

{
    "ShardId": "shardId-000000000012",
    "SequenceNumber": "49627894338614655560500811028721934184977530127978070210"
}
{
    "ShardId": "shardId-000000000012",
    "SequenceNumber": "49627894338614655560500811028795678659974022576354623682"
}
{
    "ShardId": "shardId-000000000014",
    "SequenceNumber": "49627894338659257050897872275134360684221592378842022114"
}
Wed Mar 23 11:19:57 CET 2022

$ sleep 10
$ aws kinesis put-record --stream-name hp-16 --partition-key 4 --data NA==
$ aws kinesis put-record --stream-name hp-16 --partition-key 5 --data NQ==
$ date

{
    "ShardId": "shardId-000000000010",
    "SequenceNumber": "49627894338570054070103749783042116732419934393936642210"
}
{
    "ShardId": "shardId-000000000014",
    "SequenceNumber": "49627894338659257050897872275659034489934342334017700066"
}
Wed Mar 23 11:20:10 CET 2022

$ sleep 10
$ aws kinesis put-record --stream-name hp-16 --partition-key 6 --data Ng==
$ date

{
    "ShardId": "shardId-000000000001",
    "SequenceNumber": "49627894338369347363316974173886988345467035365375213586"
}
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": "49627894338525452579706688535878947299195189349725503618"
}
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": "49627894338614655560500811029600823255837371928900796610"
}
Wed Mar 23 11:21:27 CET 2022
```

此輸入應該會產生 5 個工作階段視窗：事件 1、2、3；事件 4、5；事件 6；事件 7；事件 8。但是，該程式只產生了前 4 個視窗。

```
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:
 49627894338480851089309627289524549239292625588395704418,}}'}, 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:
 49627894338480851089309627289524549239292625588395704418,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,530 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 6 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000007,HashKeyRange: {StartingHashKey:
 148873535527910577765226390751398592512,EndingHashKey:
 170141183460469231731687303715884105727},SequenceNumberRange: {StartingSequenceNumber:
```

```
 496278943385031518345081579126660849575652739499016848 50,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 6 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000010,HashKeyRange: {StartingHashKey:
 212676479325586539664609129644855132160,EndingHashKey:
 233944127258145193631070042609340645375},SequenceNumberRange: {StartingSequenceNumber:
 496278943385700540701037497820906921123832190344196261 46,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 6 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000007,HashKeyRange: {StartingHashKey:
 148873535527910577765226390751398592512,EndingHashKey:
 170141183460469231731687303715884105727},SequenceNumberRange: {StartingSequenceNumber:
 496278943385031518345081579126660849575652739499016848 50,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,531 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 4 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000005,HashKeyRange: {StartingHashKey:
 106338239662793269832304564822427566080,EndingHashKey:
 127605887595351923798765477786913079295},SequenceNumberRange: {StartingSequenceNumber:
 496278943384585503441110966663830135210199772268897239 86,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 4 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000005,HashKeyRange: {StartingHashKey:
 106338239662793269832304564822427566080,EndingHashKey:
 127605887595351923798765477786913079295},SequenceNumberRange: {StartingSequenceNumber:
 496278943384585503441110966663830135210199772268897239 86,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 3 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000004,HashKeyRange: {StartingHashKey:
 850705917302346158658436518579420528 64,EndingHashKey:
 106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
 496278943384362495989125660432414778027473288653837435 54,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 2 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000003,HashKeyRange: {StartingHashKey:
 638029437976759618993827388934565396 48,EndingHashKey:
 850705917302346158658436518579420528 63},SequenceNumberRange: {StartingSequenceNumber:
```

```
 49627894338413948853714035420099942084474680503877763122,}}'}, 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:
 31901471898837980949691369446728269824 0,EndingHashKey:
 34028236692093846346337460743176821145 5},SequenceNumberRange: {StartingSequenceNumber:
 49627894338681557796096402897798370703746460841949528306,}}'}, 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:
 29774707105582115553045278150279718502 4,EndingHashKey:
 31901471898837980949691369446728269823 9},SequenceNumberRange: {StartingSequenceNumber:
 49627894338659257050897872274656834985473812480443547874,}}'}, 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:
 85070591730234615865843651857942052864,EndingHashKey:
 10633823966279326983230456482242756607 9},SequenceNumberRange: {StartingSequenceNumber:
 49627894338436249598912566043241477802747328865383743554,}}'} 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:
 49627894338413948853714035420099942084474680503877763122,}}'} 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:
 49627894338369347363316974173816870647929383780865802258,}}'}, 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:
```

```
 496278943385477533249052191589491563941105706729136457l4,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
 212676479325586539664609129644855l3215},SequenceNumberRange: {StartingSequenceNumber:
 496278943383470466181184435506753349296567354193598218l6,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000012,HashKeyRange: {StartingHashKey:
 255211775190703847597530955573826158592,EndingHashKey:
 276479423123262501563991868538311671807},SequenceNumberRange: {StartingSequenceNumber:
 496278943386146555605008110283737635489285157574315870l0,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000008,HashKeyRange: {StartingHashKey:
 170141183460469231731687303715884105728,EndingHashKey:
 191408831393027885698148216680369618943},SequenceNumberRange: {StartingSequenceNumber:
 496278943385254525797066885358076206758379223114076652l2,}}'}, 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:
 212676479325586539664609129644855l3216,EndingHashKey:
 425352958651173079329218259289710264l1},SequenceNumberRange: {StartingSequenceNumber:
 496278943383693473633169741738168706479293837808658022581,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000011,HashKeyRange: {StartingHashKey:
 233944127258145193631070042609340645376,EndingHashKey:
 255211775190703847597530955573826158591},SequenceNumberRange: {StartingSequenceNumber:
 496278943385923548153022804052322278306558673959256065l8,}}'}, 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:
 212676479325586539664609129644855l3215},SequenceNumberRange: {StartingSequenceNumber:
 496278943383470466181184435506753349296567354193598218l6,}}'} 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:
 49627894338391648108515504796958406366202032142371782690,}}'}, 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:
 27647942312326250156399186853831167180 8,EndingHashKey:
 29774707105582115553045278150279718502 3},SequenceNumberRange: {StartingSequenceNumber:
 49627894338636956305699341651515299267201164118937567442,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,568 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 1 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000002,HashKeyRange: {StartingHashKey:
 42535295865117307932921825928971026432,EndingHashKey:
 63802943797675961899382738893456539647},SequenceNumberRange: {StartingSequenceNumber:
 49627894338391648108515504796958406366202032142371782690,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:23,209 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000009,HashKeyRange: {StartingHashKey:
 19140883139302788569814821668036961894 4,EndingHashKey:
 21267647932558653966460912964485513215 9},SequenceNumberRange: {StartingSequenceNumber:
 49627894338547753324905219158949156394110570672913645714,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,244 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 6 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000010,HashKeyRange: {StartingHashKey:
 21267647932558653966460912964485513216 0,EndingHashKey:
 23394412725814519363107004260934064537 5},SequenceNumberRange: {StartingSequenceNumber:
 49627894338570054070103749782090692112383219034419626146,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
event: 6; timestamp: 1648030822428, 2022-03-23T10:20:22.428Z
11:59:23,377 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 3 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000015,HashKeyRange: {StartingHashKey:
 31901471898837980949691369446728269824 0,EndingHashKey:
```

```
 340282366920938463463374607431768211455},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:
 297747071055821155530452781502797185024,EndingHashKey:
 319014718988379809496913694467282698239},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:
 170141183460469231731687303715884105728,EndingHashKey:
 191408831393027885698148216680369618943},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:
 276479423123262501563991868538311671808,EndingHashKey:
 297747071055821155530452781502797185023},SequenceNumberRange: {StartingSequenceNumber:
 496278943386369563056993416515152992672011641189375674 42,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:24,790 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000012,HashKeyRange: {StartingHashKey:
 255211775190703847597530955573826158592,EndingHashKey:
 276479423123262501563991868538311671807},SequenceNumberRange: {StartingSequenceNumber:
 496278943386146555605008110283737635489285157574315870 10,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 2
event: 4; timestamp: 1648030809282, 2022-03-23T10:20:09.282Z
event: 3; timestamp: 1648030797697, 2022-03-23T10:19:57.697Z
event: 5; timestamp: 1648030810871, 2022-03-23T10:20:10.871Z
11:59:24,907 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000011,HashKeyRange: {StartingHashKey:
 233944127258145193631070042609340645376,EndingHashKey:
 255211775190703847597530955573826158591},SequenceNumberRange: {StartingSequenceNumber:
```

```
 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
```

輸出僅顯示 4 個視窗 (缺少包含事件 8 的最後一個視窗)。這是由於事件時間和浮水印策略所導致。最後一個視窗無法關閉，因為使用預先建置的浮水印策略，時間永遠不會超過從串流中讀取的最後一個事件的時間。但是對於要關閉的視窗，時間需要在最後一個事件發生後提前超過 10 秒。在這種情況下，最後一個浮水印是 2022-03-23T10:21:27.170 Z，但為了關閉該工作階段視窗，需要在 10 秒和 1 毫秒後新增浮水印。

如果從浮水印策略中刪除 `withIdleness` 選項，則不會關閉任何工作階段視窗，因為視窗運算子的「全域浮水印」無法提前。

請注意，當 Flink 應用程式啟動 (或者如果有資料扭曲) 時，某些碎片可能會比其他碎片更快地被取用。這可能會導致某些浮水印從子任務中太早發出 (子任務可能會根據一個碎片的內容發出浮水印，而不會從其訂閱的其他碎片中取用)。緩解方法是使用不同的浮水印策略，新增安全緩衝區 (`forBoundedOutOfOrderness(Duration.ofSeconds(30))` 或明確允許延遲到達的事件 (`allowedLateness(Time.minutes(5))`。

為所有運算子設定 UUID

當 Managed Service for Apache Flink 使用快照為應用程式啟動 Flink 作業時，Flink 作業可能會因為某些問題而無法啟動。其中之一是運算子 ID 不符。Flink 預期 Flink 作業圖表運算子具有明確且一致的運算子 ID。如果未明確設定，Flink 會自動產生運算子 ID。這是因為，Flink 使用這些運算子 ID 來唯一識別作業圖表中的運算子，並使用它們將每個運算子的狀態儲存在儲存點中。

當 Flink 在作業圖表的運算子 ID 與儲存點中定義的運算子 ID 之間找不到 1:1 對應時，就會發生運算子 ID 不符問題。當未設定明確一致的運算子 ID，且 Flink 自動產生的運算子 ID 可能與每個作業圖標建立的 ID 不一致時，就會發生這種情況。在維護執行時間間，應用程式遇到此問題的可能性很高。為了避免這種情況，我們建議客戶在 flink 程式碼中為所有運算子設定 UUID。如需詳細資訊，請參閱生產就緒性下的為所有運算子設定 UUID 主題。

添加 ServiceResourceTransformer 到 Maven 陰影插件

Flink 使用 Java 的服務提供者介面 (SPI) 來載入連接器和格式等元件。使用 SPI 的多個 Flink 相依性可能會在 uber-jar 中造成衝突以及非預期的應用程式行為。建議添加在 pom.xml 中定義ServiceResourceTransformer的 Maven 陰影插件

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
                <executions>
                    <execution>
                        <id>shade</id>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                        <configuration>
                            <transformers combine.children="append">
                                <!-- The service transformer is needed to merge META-
INF/services files -->
                                <transformer
 implementation="org.apache.maven.plugins.shade.resource.ServicesResourceTransformer"/>
                                <!-- ... -->
                            </transformers>
                        </configuration>
                    </execution>
                </executions>
        </plugin>
```

Apache Flink 有狀態函數

有狀態函數是一種可簡化建置分佈式有狀態應用程式的 API。它基於具有持久狀態的函數，這種函數可以與強大的一致性保證進行動態互動。

有狀態函數應用程式基本上就是一個 Apache Flink 應用程式，因此可以部署到 Managed Service for Apache Flink。不過，為 Kubernetes 叢集與為 Managed Service for Apache Flink 封裝有狀態函數之間有幾個差異。有狀態函數應用程式的最重要方面是模塊組態包含設定有狀態函數執行時間所需的所有運行時間資訊。此組態通常封裝到有狀態函數特定的容器中，並部署到 Kubernetes 上。但是，Managed Service is Apache Flink 無法如此。

以下是為 Managed Service for Apache Flink 進行的 StateFun Python 範例調整：

Apache Flink 應用程式範本

客戶可以編譯只調用有狀態函數執行時間並且包含所需相依性的 Flink 應用程式 jar，而不是使用客戶容器作為有狀態函數執行時間。對於 Flink 1.13，所需的相依項如下所示：

```
<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>
```

Flink 應用程式調用有狀態函數執行時間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);
}
```

請注意，這些元件是通用的，獨立於在有狀態函數中實作的邏輯。

模組組態的位置

有狀態函數模塊組態需要包含在類別路徑中，才能讓有狀態函數執行時間發現。最好將其包含在 Flink 應用程式的資源資料夾中，並封裝到 jar 檔案中。

與常見的 Apache Flink 應用程式類似，您然後可以使用 maven 來建立 uber jar 檔案，並將其部署到 Managed Service for Apache Flink 上。

Managed Service for Apache Flink 舊版資訊

本主題包含將 Managed Service for Apache Flink 與舊版 Apache Flink 搭配使用的相關資訊。Managed Service for Apache Flink 支援的 Apache Flink 版本包括 1.15.2 (建議使用)、1.13.2、1.11.1、1.8.2 和 1.6.2。

建議將 Apache Flink 應用程式的最新支援版本與 Managed Service for Apache Flink 搭配使用。Apache Flink 1.15.2 版具有以下功能：

- 支援 Apache Flink 資料表 API & SQL
- 支援 Python 應用程式
- 支援 Java 版本 11 和任何 Scala 版本
- 改進的記憶體模型
- RockSDB 最佳化，可提升應用程式穩定性
- Apache Flink 儀表板現支援任務管理員和堆疊追蹤。

本主題包含下列章節：

將 Apache Flink Kinesis 串流連接器與 Apache Flink 先前版本搭配使用

1.11 版之前的 Apache Flink 中不包含 Apache Flink Kinesis 串流連接器。若要讓應用程式能夠將 Apache Flink Kinesis 連接器與先前版本的 Apache Flink 搭配使用，必須下載、編譯並安裝該應用程式

所使用的 Apache Flink 版本。此連接器用於取用作為應用程式來源的 Kinesis 串流中的資料，或將資料寫入作為應用程式輸出的 Kinesis 串流。

> **Note**
>
> 確保正在使用 KPL 0.14.0 版本或更高版本建置連接器。

若要下載並安裝 Apache Flink 1.8.2 版來源程式碼，請執行下列動作：

1. 確保已安裝 Apache Maven，並且 JAVA_HOME 環境變數指向 JDK 而不是 JRE。您可以使用以下命令來測試 Apache Maven 安裝：

   ```
   mvn -version
   ```

2. 下載 Apache Flink 版本 1.8.2 來源程式碼：

   ```
   wget https://archive.apache.org/dist/flink/flink-1.8.2/flink-1.8.2-src.tgz
   ```

3. 解壓縮 Apache Flink 來源程式碼：

   ```
   tar -xvf flink-1.8.2-src.tgz
   ```

4. 切換到 Apache Flink 來源程式碼目錄：

   ```
   cd flink-1.8.2
   ```

5. 編譯並安裝 Apache Flink：

   ```
   mvn clean install -Pinclude-kinesis -DskipTests
   ```

   > **Note**
   >
   > 如果您在 Microsoft 視窗中編譯 Flink，則需要添加 `-Drat.skip=true` 參數。

建置使用 Apache Flink 1.8.2 的應用程式

本節包含您用來建置與 Apache Flink 1.8.2 搭配使用的 Managed Service for Apache Flink 之元件的相關資訊。

將下列元件版本用於 Managed Service for Apache Flink 應用程式：

元件	版本
Java	1.8 (建議使用)
Apache Flink	1.8.2
用於 Flink 執行時間 (aws-kinesisanalytics-runtime) 的 Managed Service for Apache Flink	1.0.1
Managed Service for Apache Flink Flink 連接器 (aws-kinesisanalytics-flink)	1.0.1
Apache Maven	3.1

若要編譯使用 Apache Flink 1.8.2 的應用程式，請使用下列參數執行 Maven：

```
mvn package -Dflink.version=1.8.2
```

如需使用 Apache Flink 1.8.2 版的 Managed Service for Apache Flink 應用程式的 `pom.xml` 檔案範例，請參閱 Managed Service for Apache Flink 1.8.2 入門。

如需如何為 Managed Service for Apache Flink 應用程式建置及使用應用程式程式碼的相關資訊，請參閱建立應用程式。

建置使用 Apache Flink 1.6.2 的應用程式

本節包含您用來建置與 Apache Flink 1.6.2 搭配使用的 Managed Service for Apache Flink 之元件的相關資訊。

將下列元件版本用於 Managed Service for Apache Flink 應用程式：

元件	版本
Java	1.8 (建議使用)
AWS Java 開發套件	1.11.379
Apache Flink	1.6.2
用於 Flink 執行時間 (aws-kinesisanalytics-runtime) 的 Managed Service for Apache Flink	1.0.1
Managed Service for Apache Flink Flink 連接器 (aws-kinesisanalytics-flink)	1.0.1
Apache Maven	3.1
Apache Beam	不支援用於 Apache Flink 1.6.2。

> **Note**
>
> 使用 Managed Service for Apache Flink 執行時間 1.0.1 版時，可以在 `pom.xml` 檔案中指定 Apache Flink 的版本，而不是在編譯應用程式程式碼時使用 `-Dflink.version` 參數。

如需使用 Apache Flink 1.6.2 版的 Managed Service for Apache Flink 應用程式的 `pom.xml` 檔案範例，請參閱 Managed Service for Apache Flink 1.6.2 入門。

如需如何為 Managed Service for Apache Flink 應用程式建置及使用應用程式程式碼的相關資訊，請參閱建立應用程式。

升級應用程式

若要升級 Managed Service for Apache Flink 版本，必須更新應用程式程式碼、刪除先前的應用程式，然後使用更新的程式碼建立新的應用程式。若要進行此操作，請執行下列動作：

- 將應用程式 `pom.xml` 檔案中 Managed Service for Apache Flink 執行階段和 Apache Flink Flink 連接器 (aws-kinesisanalytics-flink) 的版本變更為 1.1.0。

- 從應用程式 pom.xml 檔案中移除 flink.version 屬性。您在下一步驟中編譯應用程式程式碼時，將提供此參數。
- 使用以下命令重新編譯應用程式程式碼：

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

- 刪除現有的應用程式。再次建立應用程式，然後為應用程式的執行階段選擇 Apache Flink 1.15.2 版 (建議版本)。

> **Note**
>
> 您無法使用舊版應用程式的快照。

Apache Flink 1.6.2 和 1.8.2 中的可用連接器

Apache Flink 架構包含用於存取各種來源之資料的連接器。

- 如需 Apache Flink 1.6.2 架構中可用連接器的相關資訊，請參閱《Apache Flink 文件 (1.6.2)》https://ci.apache.org/projects/flink/flink-docs-release-1.6/中的連接器 (1.6.2)。
- 如需 Apache Flink 1.8.2 架構中可用連接器的相關資訊，請參閱《Apache Flink 文件 (1.8.2)》https://ci.apache.org/projects/flink/flink-docs-release-1.8/ 中的連接器 (1.8.2)。

Flink 1.13.2 入門

本節將為您介紹適用於 Apache Flink 的受管理服務和 DataStream API 的基本概念。它描述了建立和測試應用程式的可用選項。此外，它還提供了相關指示，以協助您安裝完成本指南教學課程以及建立您的第一個應用程式所需要的工具。

主題

- 步驟 3：建立並執行 Managed Service for Apache Flink 應用程式
- 步驟 4：清除 AWS 資源
- 步驟 5：後續步驟

Managed Service for Apache Flink 應用程式的元件

為了處理資料，您的 Managed Service for Apache Flink 使用 Java/Apache Maven 或 Scala 應用程式來處理輸入，並使用 Apache Flink 執行時間生成輸出。

Managed Service for Apache Flink 應用程式包含以下元件：

- 執行時間屬性：您可以使用執行時間屬性來設定應用程式，無需重新編譯應用程式的程式碼。
- 來源：應用程式使用來源來消耗資料。來源連接器從 Kinesis 資料串流、Amazon S3 儲存貯體等讀取資料。如需詳細資訊，請參閱來源。
- 運算子：應用程式會使用一或多個運算子來處理資料。運算子可以轉換、富集或彙總資料。如需詳細資訊，請參閱DataStream API 運算子。
- 接收器：應用程式透過使用接收器生成資料到外部來源。接收器連接器會將資料寫入 Kinesis 資料串流、Kinesis Data Firehose 串流、Amazon S3 儲存貯體等。如需詳細資訊，請參閱接收。

建立、編譯和封裝應用程式的程式碼後，將程式碼套件上傳到 Amazon Simple Storage Service (Amazon S3) 儲存貯體。然後建立 Managed Service for Apache Flink 應用程式。您可以傳入程式碼套件位置，Kinesis 資料串流作為串流資料來源，以及通常是接收應用程式處理後的資料的串流或檔案位置。

完成練習的先決條件

若要完成本指南中的步驟，您必須執行下列各項：

- Java 開發套件 (JDK) 版本 11。將 `JAVA_HOME` 環境變數設為指向您的 JDK 安裝位置。
- 我們建議您使用開發環境 (如 Eclipse Java Neon 或 IntelliJ Idea) 來開發和編譯您的應用程式。
- Git 用戶端。如果您尚未安裝 Git 用戶端，請先完成安裝。
- Apache Maven 編譯器外掛程式。Maven 必須在您的工作路徑中。若要測試您的 Apache Maven 安裝，輸入以下資訊：

```
$ mvn -version
```

開始執行，請移至 步驟 1：設定 AWS 帳戶並建立管理員使用者。

步驟 1：設定 AWS 帳戶並建立管理員使用者

註冊 AWS 帳戶

如果您還沒有 AWS 帳戶，請完成以下步驟建立新帳戶。

註冊 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   註冊 AWS 帳戶時，會建立 AWS 帳戶根使用者。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為最佳安全實務，將管理存取權指派給管理使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

註冊程序完成後，AWS 會傳送一封確認電子郵件給您。您可以隨時登錄 https://aws.amazon.com/ 並選擇 我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立管理使用者

當您註冊 AWS 帳戶 之後，請保護您的 AWS 帳戶根使用者，啟用 AWS IAM Identity Center，並建立管理使用者，讓您可以不使用根使用者處理日常作業。

保護您的 AWS 帳戶根使用者

1. 選擇 根使用者 並輸入您的 AWS 帳戶電子郵件地址，以帳戶擁有者身分登入 AWS Management Console。在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入使用者指南中的以根使用者身分登入。
2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶根使用者啟用虛擬 MFA 裝置 (主控台)。

建立管理使用者

1. 啟用 IAM Identity Center。

如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，將管理權限授予管理使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的教學課程，請參閱《使用 AWS IAM Identity Center 使用者指南中的以預設 IAM Identity Center 目錄 設定使用者存取權限。

以管理員的身分登入

- 若要使用您的 IAM 身分中心使用者登入，請使用建立 IAM 身分中心使用者時傳送至您電子郵件地址的登入 URL。

  如需有關如何使用 IAM Identity Center 使用者登入的說明，請參閱《AWS 登入 使用者指南》中的登入 AWS存取入口網站。

授予程式設計存取權

若使用者想要與 AWS Management Console 之外的 AWS 互動，則需要程式設計存取權。授予程式設計存取權的方式取決於存取 AWS 的使用者類型。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 設定 AWS CLI 來使用 AWS IAM Identity Center。 • 關於 AWS SDKs、工具和 AWS APIs，請參閱 AWSSDKs 和工具參考指南 中的 IAM Identity Center 驗證。

哪個使用者需要程式設計存取權？	到	By
IAM	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請遵循 IAM 使用者指南 中 使用臨時憑證搭配 AWS 資源 中的指示。
IAM	(不建議使用) 使用長期憑證簽署 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 使用 IAM 使用者憑證進行驗證。 • 關於 AWS SDKs 和工具，請參閱 AWSSDKs 和工具參考指南 中的 使用長期憑證進行驗證。 • 關於 AWS API，請參閱 IAM 使用者指南 中的 管理 IAM 使用者的存取金鑰。

後續步驟

步驟 2：設定 AWS Command Line Interface (AWS CLI)

後續步驟

步驟 2：設定 AWS Command Line Interface (AWS CLI)

步驟 2：設定 AWS Command Line Interface (AWS CLI)

在此步驟中，您將下載並設定與 Amazon Managed Service for Apache Flink 搭配使用的 AWS CLI。

> **Note**
>
> 本指南的入門練習均假設您使用帳戶中的管理員登入資料 (adminuser) 來執行操作。

> **Note**
>
> 如果您已經安裝 AWS CLI，則可能需要升級才能取得最新功能。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的安裝 AWS Command Line Interface。若要查看 AWS CLI 的版本，執行以下命令：
>
> ```
> aws --version
> ```
>
> 此教學課程中的練習需要以下 AWS CLI 版本或更新版本：
>
> ```
> aws-cli/1.16.63
> ```

設定 AWS CLI

1. 下載和設定 AWS CLI。如需說明，請參閱《AWS Command Line Interface 使用者指南》中的下列主題：
   - 安裝 AWS Command Line Interface
   - 設定 AWS CLI
2. 在 AWS CLIconfig 檔案中，為管理員使用者新增具名設定檔。當您執行 AWS CLI 命令時，使用此設定檔。如需具名描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的具名描述檔。

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

   如需可用 AWS 區域的清單，請參閱 Amazon Web Services 一般參考 中的 區域與端點。

> **Note**
>
> 本教學課程中的範例程式碼和命令使用美國西部 (奧勒岡) 區域。若要使用其他區域，請將本教學課程的程式碼和指令中的「區域」變更為您要使用的區域。

3. 在命令提示字元中輸入下列 help 命令，以驗證設定：

```
aws help
```

設定AWS帳戶之後AWS CLI，您可以嘗試下一個練習，在其中設定範例應用程式並測試 end-to-end 設定。

後續步驟

步驟 3：建立並執行 Managed Service for Apache Flink 應用程式

步驟 3：建立並執行 Managed Service for Apache Flink 應用程式

在本練習中，您會建立 Managed Service for Apache Flink 應用程式，並將資料串流作為來源和目的地。

本節包含下列步驟：

- 建立兩個 Amazon Kinesis Data Streams
- 寫入範例記錄至輸入串流
- 下載並檢查 Apache Flink 串流 Java 程式碼
- 編譯應用程式的程式碼
- 上傳 Apache Flink 串流 Java 程式碼
- 建立並執行 Managed Service for Apache Flink 應用程式
- 後續步驟

建立兩個 Amazon Kinesis Data Streams

在為本練習建立 Managed Service for Apache Flink 應用程式之前，請先建立兩個 Kinesis 資料串流 (`ExampleInputStream` 和 `ExampleOutputStream`)。您的應用程式會將這些串流用於應用程式來源和目的地串流。

您可以使用 Amazon Kinesis 主控台或以下 AWS CLI 命令來建立這些串流。如需主控台指示，請參閱《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。

建立資料串流 (AWS CLI)

1. 若要建立第一個串流 (ExampleInputStream)，請使用以下 Amazon Kinesis `create-stream` AWS CLI 命令。

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

2. 若要建立應用程式用來寫入輸出的第二個串流，請執行相同的命令，將串流名稱變更為 `ExampleOutputStream`。

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

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

1. 使用下列內容建立名為 `stock.py` 的檔案：

```
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. 在教學課程後半段，您會執行 `stock.py` 指令碼來傳送資料至應用程式。

```
$ python stock.py
```

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的 Java 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，執行下列操作：

1. 使用以下指令複製遠端儲存庫：

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

2. 請前往 `amazon-kinesis-data-analytics-java-examples/GettingStarted` 目錄。

請留意下列與應用程式的程式碼相關的資訊：

- 專案物件模型 (pom.xml) 檔案包含應用程式的組態和相依性資訊，包括 Managed Service for Apache Flink 程式庫。
- `BasicStreamingJob.java` 檔案包含定義應用程式功能的 `main` 方法。
- 應用程式使用 Kinesis 來源從來源串流讀取。以下程式碼片段會建立 Kinesis 來源：

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

- 您的應用程式會建立來源與目的地連接器，以使用 StreamExecutionEnvironment 物件來存取外部資源。
- 應用程式會使用靜態屬性來建立來源與目的地連接器。若要使用動態應用程式屬性，請使用 createSourceFromApplicationProperties 和 createSinkFromApplicationProperties 方法來建立連接器。這些方法會讀取應用程式的屬性，來設定連接器。

  如需執行時間屬性的詳細資訊，請參閱執行時間屬性。

編譯應用程式的程式碼

在本節中，您會使用 Apache Maven 編譯器來建立應用程式的 Java 程式碼。如需安裝 Apache Maven 和 Java 開發套件 (JDK) 的詳細資訊，請參閱 完成練習的先決條件。

編譯應用程式的程式碼

1. 請將應用程式的程式碼編譯並封裝成 JAR 檔案，以使用應用程式的程式碼。您可以使用下列兩種方式的其中之一，編譯和封裝您的程式碼：

   - 使用命令列 Maven 工具。請在包含 pom.xml 檔案的目錄中執行下列命令，來建立 JAR 檔案：

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

   - 設定開發環境。如需詳細資訊，請參閱您的開發環境文件。

     > **Note**
     >
     > 提供的來源程式碼依賴於 Java 11 中的程式庫。

   您可以將您的套件做為 JAR 檔案上傳，或壓縮您的套件並做為 ZIP 檔案上傳。如果是使用 AWS CLI 建立應用程式，您會指定程式碼內容類型 (JAR 或 ZIP)。
2. 如果編譯時發生錯誤，請確認您的 JAVA_HOME 環境變數是否正確設定。

如果應用程式成功編譯，則會建立下列檔案：

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

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會建立 Amazon Simple Storage Service (Amazon S3) 儲存貯體並上傳您的應用程式的程式碼。

上傳應用程式的程式碼

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 選擇建立儲存貯體。
3. 在儲存貯體名稱欄位中，輸入 **ka-app-code-*<username>***。新增尾碼至儲存貯體名稱，例如您的使用者名稱，使其成為全域唯一的。選擇下一步。
4. 在設定選項步驟中，保留原有設定並選擇 Next (下一步)。
5. 在設定許可步驟中，保留原有設定並選擇 Next (下一步)。
6. 選擇建立儲存貯體。
7. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
8. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `aws-kinesis-analytics-java-apps-1.0.jar` 檔案。選擇下一步。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

您可以使用主控台或 AWS CLI 建立和執行 Managed Service for Apache Flink 應用程式。

> **Note**
>
> 使用主控台建立應用程式時，系統會為您建立 AWS Identity and Access Management (IAM) 和 Amazon CloudWatch 日誌資源。當您使用 AWS CLI 建立應用程式時，則會個別建立這些資源。

主題

- 建立和執行應用程式 (主控台)

- 建立並執行應用程式 (AWS CLI)

建立和執行應用程式 (主控台)

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 對於 Description (說明)，輸入 **My java test app**。
   - 針對執行時間，選擇 Apache Flink。
   - 將版本下拉式清單保留為 Apache Flink 1.13 版。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：`kinesis-analytics-service-MyApplication-us-west-2`
> - 角色：`kinesisanalytics-MyApplication-us-west-2`

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。

3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。

4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **aws-kinesis-analytics-java-apps-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 輸入下列資料：

群組 ID	金鑰	值
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**

群組 ID	金鑰	值
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

5. 在監控下，確保監控指標層級設為應用程式。
6. 若要CloudWatch 記錄，請選取 [啟用] 核取方塊。
7. 選擇更新。

> **Note**
>
> 當您選擇啟用 Amazon 日誌 CloudWatch 記錄時，Apache Flink 的受管服務會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

停止應用程式

在MyApplication頁面上，選擇 [停止]。確認動作。

更新應用程式

您可以使用主控台更新應用程式設定，例如應用程式屬性、監控設定及位置或應用程式 JAR 的檔名。如果需要更新應用程式的程式碼，也可以從 Amazon S3 儲存貯體重新載入應用程式 JAR。

在MyApplication頁面上，選擇設定。更新應用程式設定，然後選擇更新。

建立並執行應用程式 (AWS CLI)

在本節中，您可以使用 AWS CLI 建立和執行 Managed Service for Apache Flink 應用程式。Managed Service for Apache Flink 使用 `kinesisanalyticsv2` AWS CLI 命令建立 Managed Service for Apache Flink 應用程式並與之互動。

建立許可政策

> **Note**
>
> 您必須為應用程式建立許可政策和角色。如果您未建立這些 IAM 資源，應用程式將無法存取其資料和日誌串流。

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 `read` 動作的許可，而另一條則是授與目的地串流上 `write` 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

使用以下程式碼來建立 `AKReadSourceStreamWriteSinkStream` 許可政策。以您用於建立 Amazon S3 儲存貯體 (以儲存應用程式的程式碼) 的使用者名稱來取代 *`username`*。以您的帳戶 ID 取代 Amazon Resource Names (ARNs) (*`012345678901`*) 中的帳戶 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"
        }
    ]
}
```

如需建立許可政策的指 step-by-step 示，請參閱 IAM 使用指南中的教學課程：建立和附加您的第一個客戶受管政策。

> **Note**
>
> 若要存取其他 Amazon 服務，您可以使用 AWS SDK for Java。Managed Service for Apache Flink 自動將 SDK 所需的憑證設定為與應用程式相關聯的服務執行 IAM 角色。無須採取額外的步驟。

建立 IAM 角色

在本節中，您會建立 Managed Service for Apache Flink 應用程式可以擔任的 IAM 角色，以便讀取來源串流與寫入目的地串流。

Managed Service for Apache Flink 沒有許可，無法存取串流。您可以透過 IAM 角色來授與這些許可。各 IAM 角色都有連接兩項政策。信任政策會授與擔任角色的 Managed Service for Apache Flink 許可，而許可政策決定了 Managed Service for Apache Flink 在擔任角色後可以執行的作業。

您會將在上一節中建立的許可政策連接至此角色。

若要建立一個 IAM 角色

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Roles (角色) 、Create Role (建立角色)。
3. 在選取可信身分類型下，選擇 AWS 服務。在 Choose the service that will use this role (選擇將使用此角色的服務) 下，選擇 Kinesis。在 Select your use case (選取您的使用案例) 下，選擇 Kinesis Analytics (Kinesis 分析)。

選擇 Next: Permissions (下一步：許可)。

4. 在 Attach permissions policies (連接許可政策) 頁面上，選擇 Next: Review (下一步：檢閱)。您會在建立角色後連接許可政策。
5. 在建立角色頁面，輸入 **MF-stream-rw-role** 作為角色名稱。選擇建立角色。

   現在您已建立新的 IAM 角色，名為 MF-stream-rw-role。您接著會更新角色的信任和許可政策。
6. 將 許可政策連接到角色。

   > **Note**
   >
   > 在此練習中，Managed Service for Apache Flink 擔任從 Kinesis 資料串流 (來源) 讀取資料並將輸出寫入另一個 Kinesis 資料串流的角色。因此您會連接在上一個步驟中建立的政策，the section called “建立許可政策”。

   a. 在摘要頁面上，選擇許可標籤。
   b. 選擇 Attach Policies (連接政策)。
   c. 在搜尋方塊中，輸入 **AKReadSourceStreamWriteSinkStream** (您在上一節中建立的政策)。
   d. 選擇 AK 原ReadSourceStreamWriteSinkStream則，然後選擇 [附加原則]。

您現在已建立應用程式用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的指 step-by-step 示，請參閱 IAM 使用者指南中的建立 IAM 角色 (主控台)。

建立 Managed Service for Apache Flink 應用程式

1. 將下列 JSON 程式碼複製到名為 create_request.json 的檔案。以您之前建立之角色的 ARN，取代範例角色 ARN。以您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (*username*)。以您的帳戶 ID 取代服務執行角色中的範例帳戶 ID (*012345678901*)。

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-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. 以建立應用程式的上述請求，執行 CreateApplication 動作：

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

現在已建立應用程式。您會在下一個步驟中啟動應用程式。

啟動應用程式

在本節中，您會透過 StartApplication 動作來啟動應用程式。

啟動應用程式

1. 將下列 JSON 程式碼複製到名為 `start_request.json` 的檔案。

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

2. 以啟動應用程式的上述請求，執行 StartApplication 動作：

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

應用程式現在正在執行。您可以在 Amazon CloudWatch 主控台上查看適用於 Apache Flink 的受管服務指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 StopApplication 動作來停止應用程式。

停止應用程式

1. 將下列 JSON 程式碼複製到名為 `stop_request.json` 的檔案。

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

2. 以停止應用程式的上述請求，執行 StopApplication 動作：

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

現在已停止應用程式。

新增記 CloudWatch 錄選項

您可以使用AWS CLI將 Amazon CloudWatch 日誌串流新增至您的應用程式。如需搭配應用程式使用 CloudWatch 記錄的相關資訊，請參閱the section called “設定日誌”。

更新環境屬性

在本節中，您可以使用 UpdateApplication 動作來變更應用程式的環境屬性，無需重新編譯應用程式的程式碼。在此範例中，您會變更來源和目的地串流的「區域」。

更新應用程式的環境屬性

1. 將下列 JSON 程式碼複製到名為 update_properties_request.json 的檔案。

```
{"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. 使用前述請求執行 UpdateApplication 動作以更新環境屬性：

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

更新應用程式的程式碼

需要使用新版本的程式碼套件更新應用程式的程式碼時，請使用 UpdateApplicationAWS CLI 動作。

> **Note**
>
> 若要載入具有相同檔案名稱的新版應用程式的程式碼，必須指定新的物件版本。如需關於使用 Amazon S3 物件版本的詳細資訊，請參閱啟用或停用版本控制。

若要使用 AWS CLI，請從 Amazon S3 儲存貯體刪除先前的程式碼套件，上傳新版本，然後呼叫 UpdateApplication，指定相同的 Amazon S3 儲存貯體和物件名稱，以及新的物件版本。應用程式將以新的程式碼套件重新啟動。

UpdateApplication 動作的下列範例請求會重新載入應用程式的程式碼並重新啟動應用程式。將 CurrentApplicationVersionId 更新至目前的應用程式版本。您可以使用 ListApplications 或 DescribeApplication 動作來檢查目前的應用程式版本。使用您在the section called “建立兩個 Amazon Kinesis Data Streams”一節中選擇的尾碼更新儲存貯體名稱尾碼 (*<username>*)。

```
{
    "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"
                }
            }
        }
    }
}
```

後續步驟

步驟 4：清除 AWS 資源

步驟 4：清除 AWS 資源

本節包括清除在入門教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

刪除 Managed Service for Apache Flink 應用程式

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis Data Streams

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇，選擇「ExampleOutputStream動作」，選擇「刪除」，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

後續步驟

步驟 5：後續步驟

步驟 5：後續步驟

現在您已建立並執行 Managed Service for Apache Flink 應用程式，請參閱下列資源，取得更進階的 Managed Service for Apache Flink 解決方案。

- Amazon Kinesis 的 AWS 串流資料解決方案：Amazon Kinesis 的 AWS 串流資料解決方案可自動設定輕鬆擷取、存放、處理和交付串流資料所需的 AWS 服務。該解決方案提供了多種解決串流資料使用案例的選項。Apache Flink 的受管理服務選項提供 end-to-end串流 ETL 範例，展示真實世界的應用程式，該應用程式會根據模擬的紐約計程車資料執行分析作業。此解決方案會設定所有必要的 AWS資源，例如 IAM 角色和政策、 CloudWatch 儀表板和 CloudWatch 警示。
- 適用於 Amazon MSK 的 AWS 串流資料解決方案：適用於 Amazon MSK 的 AWS 串流資料解決方案提供 AWS CloudFormation 範本，可讓資料流經生產者、串流儲存、取用者和目的地。

- 使用 Apache Flink 和 Apache Kafka 的點擊流實驗室：點擊流使用案例的端對端實驗室，使用 Amazon Managed Streaming for Apache Kafka 進行串流儲存，使用適用於 Apache Flink 應用程式的 Managed Service for Apache Flink 進行串流處理。
- 適用於 Apache Flink 工作坊的 Amazon 受管服務：在本研討會中，您可以建立 end-to-end 串流架構，以近乎即時的方式擷取、分析串流資料並以視覺化方式呈現。您著手改善紐約市一家出租車公司的運營。您可以近乎即時地分析紐約市計程車車隊的遙測資料，以最佳化其車隊運作。
- Managed Service for Apache Flink：範例：本開發人員指南的這一節提供了在 Managed Service for Apache Flink 中建立和使用應用程式的範例。其中包含範例程式碼和 step-by-step 指示，可協助您為 Apache Flink 應用程式建立受管理服務並測試結果。
- 學習 Flink：動手訓練：官方介紹性 Apache Flink 訓練課程，可協助您開始撰寫可擴展的串流 ETL、分析和事件驅動型應用程式。

> **Note**
>
> 請注意，Managed Service for Apache Flink 不支援本訓練中使用的 Apache Flink 版本 (1.12)。您可以在阿帕奇 Flink 的 Flink 管理服務中使用 Flink 1.15.2。

Flink 1.11.1 入門

本主題包含使用 Apache Flink 1.11.1 的 開始使DataStream 用 (API) 教學課程版本。

本節將為您介紹適用於 Apache Flink 的受管理服務和 DataStream API 的基本概念。它描述了建立和測試應用程式的可用選項。此外，它還提供了相關指示，以協助您安裝完成本指南教學課程以及建立您的第一個應用程式所需要的工具。

主題

- Managed Service for Apache Flink 應用程式的元件
- 完成練習的先決條件
- 步驟 1：設定 AWS 帳戶並建立管理員使用者
- 步驟 2：設定 AWS Command Line Interface (AWS CLI)
- 步驟 3：建立並執行 Managed Service for Apache Flink 應用程式
- 步驟 4：清除 AWS 資源
- 步驟 5：後續步驟

Managed Service for Apache Flink 應用程式的元件

為了處理資料，您的 Managed Service for Apache Flink 應用程式使用 Java/Apache Maven 或 Scala 應用程式來處理輸入，使用 Apache Flink 執行時間生成輸出。

Managed Service for Apache Flink 應用程式包含下列元件：

- 執行時間屬性：您可以使用執行時間屬性來設定應用程式，無需重新編譯應用程式的程式碼。
- 來源：應用程式使用來源來消耗資料。來源連接器從 Kinesis 資料串流、Amazon S3 儲存貯體等讀取資料。如需詳細資訊，請參閱來源。
- 運算子：應用程式會使用一或多個運算子來處理資料。運算子可以轉換、富集或彙總資料。如需詳細資訊，請參閱DataStream API 運算子。
- 接收器：應用程式透過使用接收器生成資料到外部來源。接收器連接器會將資料寫入 Kinesis 資料串流、Kinesis Data Firehose 串流、Amazon S3 儲存貯體等。如需詳細資訊，請參閱接收。

建立、編譯和封裝應用程式的程式碼後，將程式碼套件上傳到 Amazon Simple Storage Service (Amazon S3) 儲存貯體。然後建立 Managed Service for Apache Flink 應用程式。您可以傳入程式碼套件位置，Kinesis 資料串流作為串流資料來源，以及通常是接收應用程式處理後的資料的串流或檔案位置。

完成練習的先決條件

若要完成本指南中的步驟，您必須執行下列各項：

- Java 開發套件 (JDK) 版本 11。將 `JAVA_HOME` 環境變數設為指向您的 JDK 安裝位置。
- 我們建議您使用開發環境 (如 Eclipse Java Neon 或 IntelliJ Idea) 來開發和編譯您的應用程式。
- Git 用戶端。如果您尚未安裝 Git 用戶端，請先完成安裝。
- Apache Maven 編譯器外掛程式。Maven 必須在您的工作路徑中。若要測試您的 Apache Maven 安裝，輸入以下資訊：

```
$ mvn -version
```

開始執行，請移至 步驟 1：設定 AWS 帳戶並建立管理員使用者。

步驟 1：設定 AWS 帳戶並建立管理員使用者

註冊 AWS 帳戶

如果您還沒有 AWS 帳戶，請完成以下步驟建立新帳戶。

註冊 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   註冊 AWS 帳戶時，會建立 AWS 帳戶根使用者。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為最佳安全實務，將管理存取權指派給管理使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

註冊程序完成後，AWS 會傳送一封確認電子郵件給您。您可以隨時登錄 https://aws.amazon.com/ 並選擇 我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立管理使用者

當您註冊 AWS 帳戶 之後，請保護您的 AWS 帳戶根使用者，啟用 AWS IAM Identity Center，並建立管理使用者，讓您可以不使用根使用者處理日常作業。

保護您的 AWS 帳戶根使用者

1. 選擇 根使用者 並輸入您的 AWS 帳戶電子郵件地址，以帳戶擁有者身分登入 AWS Management Console。在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入使用者指南中的以根使用者身分登入。
2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶根使用者啟用虛擬 MFA 裝置 (主控台)。

建立管理使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，將管理權限授予管理使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的教學課程，請參閱《使用 AWS IAM Identity Center 使用者指南中的以預設 IAM Identity Center 目錄 設定使用者存取權限。

以管理員的身分登入

- 若要使用您的 IAM 身分中心使用者登入，請使用建立 IAM 身分中心使用者時傳送至您電子郵件地址的登入 URL。

  如需有關如何使用 IAM Identity Center 使用者登入的說明，請參閱《AWS 登入 使用者指南》中的登入 AWS存取入口網站。

授予程式設計存取權

若使用者想要與 AWS Management Console 之外的 AWS 互動，則需要程式設計存取權。授予程式設計存取權的方式取決於存取 AWS 的使用者類型。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 設定 AWS CLI 來使用 AWS IAM Identity Center。 • 關於 AWS SDKs、工具和 AWS APIs，請參閱 AWSSDKs 和工具參考指南 中的 IAM Identity Center 驗證。

哪個使用者需要程式設計存取權？	到	By
IAM	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請遵循 IAM 使用者指南 中 使用臨時憑證搭配 AWS 資源 中的指示。
IAM	(不建議使用) 使用長期憑證簽署 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 使用 IAM 使用者憑證進行驗證。 • 關於 AWS SDKs 和工具，請參閱 AWSSDKs 和工具參考指南 中的 使用長期憑證進行驗證。 • 關於 AWS API，請參閱 IAM 使用者指南 中的 管理 IAM 使用者的存取金鑰。

後續步驟

步驟 2：設定 AWS Command Line Interface (AWS CLI)

步驟 2：設定 AWS Command Line Interface (AWS CLI)

在此步驟中，您將下載並設定與 Amazon Managed Service for Apache Flink 搭配使用的 AWS CLI。

> **Note**
>
> 本指南的入門練習均假設您使用帳戶中的管理員登入資料 (`adminuser`) 來執行操作。

> **Note**
>
> 如果您已經安裝 AWS CLI，則可能需要升級才能取得最新功能。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的安裝 AWS Command Line Interface。若要查看 AWS CLI 的版本，執行以下命令：
>
> ```
> aws --version
> ```
>
> 此教學課程中的練習需要以下 AWS CLI 版本或更新版本：
>
> ```
> aws-cli/1.16.63
> ```

設定 AWS CLI

1. 下載和設定 AWS CLI。如需說明，請參閱《AWS Command Line Interface 使用者指南》中的下列主題：
   - 安裝 AWS Command Line Interface
   - 設定 AWS CLI
2. 在 AWS CLIconfig 檔案中，為管理員使用者新增具名設定檔。當您執行 AWS CLI 命令時，使用此設定檔。如需具名描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的具名描述檔。

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

   如需可用 AWS 區域的清單，請參閱 Amazon Web Services 一般參考 中的 區域與端點。

   > **Note**
   >
   > 本教學課程中的範例程式碼和命令使用美國西部 (奧勒岡) 區域。若要使用其他區域，請將本教學課程的程式碼和指令中的「區域」變更為您要使用的區域。

3. 在命令提示字元中輸入下列 help 命令，以驗證設定：

```
aws help
```

設定AWS帳戶之後AWS CLI，您可以嘗試下一個練習，在其中設定範例應用程式並測試 end-to-end 設定。

後續步驟

步驟 3：建立並執行 Managed Service for Apache Flink 應用程式

步驟 3：建立並執行 Managed Service for Apache Flink 應用程式

在本練習中，您會建立 Managed Service for Apache Flink 應用程式，並將資料串流作為來源和目的地。

本節包含下列步驟：

- 建立兩個 Amazon Kinesis Data Streams
- 寫入範例記錄至輸入串流
- 下載並檢查 Apache Flink 串流 Java 程式碼
- 編譯應用程式的程式碼
- 上傳 Apache Flink 串流 Java 程式碼
- 建立並執行 Managed Service for Apache Flink 應用程式
- 後續步驟

建立兩個 Amazon Kinesis Data Streams

在為本練習建立 Managed Service for Apache Flink 應用程式之前，請先建立兩個 Kinesis 資料串流 (`ExampleInputStream` 和 `ExampleOutputStream`)。您的應用程式會將這些串流用於應用程式來源和目的地串流。

您可以使用 Amazon Kinesis 主控台或以下 AWS CLI 命令來建立這些串流。如需主控台指示，請參閱《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。

建立資料串流 (AWS CLI)

1. 若要建立第一個串流 (`ExampleInputStream`)，請使用以下 Amazon Kinesis `create-stream` AWS CLI 命令。

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

2. 若要建立應用程式用來寫入輸出的第二個串流，請執行相同的命令，將串流名稱變更為 ExampleOutputStream。

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

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

1. 使用下列內容建立名為 stock.py 的檔案：

```
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. 在教學課程後半段，您會執行 `stock.py` 指令碼來傳送資料至應用程式。

```
$ python stock.py
```

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的 Java 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，執行下列操作：

1. 使用以下指令複製遠端儲存庫：

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

2. 請前往 `amazon-kinesis-data-analytics-java-examples/GettingStarted` 目錄。

請留意下列與應用程式的程式碼相關的資訊：

- 專案物件模型 (pom.xml) 檔案包含應用程式的組態和相依性資訊，包括 Managed Service for Apache Flink 程式庫。
- `BasicStreamingJob.java` 檔案包含定義應用程式功能的 `main` 方法。
- 應用程式使用 Kinesis 來源從來源串流讀取。以下程式碼片段會建立 Kinesis 來源：

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

- 您的應用程式會建立來源與目的地連接器，以使用 StreamExecutionEnvironment 物件來存取外部資源。
- 應用程式會使用靜態屬性來建立來源與目的地連接器。若要使用動態應用程式屬性，請使用 createSourceFromApplicationProperties 和 createSinkFromApplicationProperties 方法來建立連接器。這些方法會讀取應用程式的屬性，來設定連接器。

  如需執行時間屬性的詳細資訊，請參閱執行時間屬性。

編譯應用程式的程式碼

在本節中，您會使用 Apache Maven 編譯器來建立應用程式的 Java 程式碼。如需安裝 Apache Maven 和 Java 開發套件 (JDK) 的詳細資訊，請參閱 完成練習的先決條件。

編譯應用程式的程式碼

1. 請將應用程式的程式碼編譯並封裝成 JAR 檔案，以使用應用程式的程式碼。您可以使用下列兩種方式的其中之一，編譯和封裝您的程式碼：

   - 使用命令列 Maven 工具。請在包含 pom.xml 檔案的目錄中執行下列命令，來建立 JAR 檔案：

     ```
     mvn package -Dflink.version=1.11.3
     ```

   - 設定開發環境。如需詳細資訊，請參閱您的開發環境文件。

     > **Note**
     >
     > 提供的來源程式碼依賴於 Java 11 中的程式庫。確保您專案的 Java 版本是 11。

   您可以將您的套件做為 JAR 檔案上傳，或壓縮您的套件並做為 ZIP 檔案上傳。如果是使用 AWS CLI 建立應用程式，您會指定程式碼內容類型 (JAR 或 ZIP)。
2. 如果編譯時發生錯誤，請確認您的 JAVA_HOME 環境變數是否正確設定。

如果應用程式成功編譯，則會建立下列檔案：

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

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會建立 Amazon Simple Storage Service (Amazon S3) 儲存貯體並上傳您的應用程式的程式碼。

上傳應用程式的程式碼

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 選擇建立儲存貯體。
3. 在儲存貯體名稱欄位中，輸入 **ka-app-code-*<username>***。新增尾碼至儲存貯體名稱，例如您的使用者名稱，使其成為全域唯一的。選擇下一步。
4. 在設定選項步驟中，保留原有設定並選擇 Next (下一步)。
5. 在設定許可步驟中，保留原有設定並選擇 Next (下一步)。
6. 選擇建立儲存貯體。
7. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
8. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `aws-kinesis-analytics-java-apps-1.0.jar` 檔案。選擇下一步。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

您可以使用主控台或 AWS CLI 建立和執行 Managed Service for Apache Flink 應用程式。

> **Note**
>
> 使用主控台建立應用程式時，系統會為您建立 AWS Identity and Access Management (IAM) 和 Amazon CloudWatch 日誌資源。當您使用 AWS CLI 建立應用程式時，則會個別建立這些資源。

主題

- 建立和執行應用程式 (主控台)

- 建立並執行應用程式 (AWS CLI)

建立和執行應用程式 (主控台)

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 對於 Description (說明)，輸入 **My java test app**。
   - 針對執行時間，選擇 Apache Flink。
   - 將版本下拉式清單保留為 Apache Flink 1.11 版 (建議版本)。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：`kinesis-analytics-service-MyApplication-us-west-2`
> - 角色：`kinesisanalytics-MyApplication-us-west-2`

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。

3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。

4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **aws-kinesis-analytics-java-apps-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 在 Properties (屬性) 下，針對 Group ID (群組 ID)，輸入 **ProducerConfigProperties**。
5. 輸入以下應用程式屬性和數值：

群組 ID	金鑰	值
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

6. 在監控下，確保監控指標層級設為應用程式。
7. 若要CloudWatch 記錄，請選取 [啟用] 核取方塊。
8. 選擇更新。

> **Note**
>
> 當您選擇啟用 Amazon 日誌 CloudWatch 記錄時，Apache Flink 的受管服務會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`

執行應用程式

透過執行應用程式、開啟 Apache Flink 儀表板並選擇所需的 Flink 作業，即可檢視 Flink 作業圖表。

停止應用程式

在MyApplication頁面上，選擇 [停止]。確認動作。

更新應用程式

您可以使用主控台更新應用程式設定，例如應用程式屬性、監控設定及位置或應用程式 JAR 的檔名。如果需要更新應用程式的程式碼，也可以從 Amazon S3 儲存貯體重新載入應用程式 JAR。

在MyApplication頁面上，選擇設定。更新應用程式設定，然後選擇更新。

建立並執行應用程式 (AWS CLI)

在本節中，您會使用 AWS CLI 建立並執行 Managed Service in Apache Flink 應用程式。Managed Service in Apache Flink 使用 `kinesisanalyticsv2` AWS CLI 命令建立 Managed Service in Apache Flink 應用程式並與之互動。

建立許可政策

> **Note**
>
> 您必須為應用程式建立許可政策和角色。如果您未建立這些 IAM 資源，應用程式將無法存取其資料和日誌串流。

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 `read` 動作的許可，而另一條則是授與目的地串流上 `write` 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

使用以下程式碼來建立 `AKReadSourceStreamWriteSinkStream` 許可政策。以您用於建立 Amazon S3 儲存貯體 (以儲存應用程式的程式碼) 的使用者名稱來取代 *`username`*。以您的帳戶 ID 取代 Amazon Resource Names (ARNs) (*`012345678901`*) 中的帳戶 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"
        }
    ]
}
```

如需建立許可政策的指 step-by-step 示，請參閱 IAM 使用指南中的教學課程：建立和附加您的第一個客戶受管政策。

> **Note**
>
> 若要存取其他 Amazon 服務，您可以使用 AWS SDK for Java。Managed Service for Apache Flink 自動將 SDK 所需的憑證設定為與應用程式相關聯的服務執行 IAM 角色。無須採取額外的步驟。

建立 IAM 角色

在本節中，您會建立 Managed Service for Apache Flink 應用程式可以擔任的 IAM 角色，以便讀取來源串流與寫入目的地串流。

Managed Service for Apache Flink 沒有許可，無法存取串流。您可以透過 IAM 角色來授與這些許可。各 IAM 角色都有連接兩項政策。信任政策會授與擔任角色的 Managed Service for Apache Flink 許可，而許可政策決定了 Managed Service for Apache Flink 在擔任角色後可以執行的作業。

您會將在上一節中建立的許可政策連接至此角色。

若要建立一個 IAM 角色

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Roles (角色) 、Create Role (建立角色)。
3. 在選取可信身分類型下，選擇 AWS 服務。在 Choose the service that will use this role (選擇將使用此角色的服務) 下，選擇 Kinesis。在 Select your use case (選取您的使用案例) 下，選擇 Kinesis Analytics (Kinesis 分析)。

選擇 Next: Permissions (下一步：許可)。

4. 在 Attach permissions policies (連接許可政策) 頁面上，選擇 Next: Review (下一步：檢閱)。您會在建立角色後連接許可政策。
5. 在建立角色頁面，輸入 **MF-stream-rw-role** 作為角色名稱。選擇建立角色。

   現在您已建立新的 IAM 角色，名為 MF-stream-rw-role。您接著會更新角色的信任和許可政策。
6. 將 許可政策連接到角色。

   > **Note**
   >
   > 在此練習中，Managed Service for Apache Flink 擔任從 Kinesis 資料串流 (來源) 讀取資料並將輸出寫入另一個 Kinesis 資料串流的角色。因此您會連接在上一個步驟中建立的政策，the section called “建立許可政策”。

   a. 在摘要頁面上，選擇許可標籤。
   b. 選擇 Attach Policies (連接政策)。
   c. 在搜尋方塊中，輸入 **AKReadSourceStreamWriteSinkStream** (您在上一節中建立的政策)。
   d. 選擇 AK 原ReadSourceStreamWriteSinkStream則，然後選擇 [附加原則]。

您現在已建立應用程式用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的指 step-by-step 示，請參閱 IAM 使用者指南中的建立 IAM 角色 (主控台)。

建立 Managed Service for Apache Flink 應用程式

1. 將下列 JSON 程式碼複製到名為 create_request.json 的檔案。以您之前建立之角色的 ARN，取代範例角色 ARN。以您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (*username*)。以您的帳戶 ID 取代服務執行角色中的範例帳戶 ID (*012345678901*)。

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_11",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-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. 以建立應用程式的上述請求，執行 CreateApplication 動作：

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

現在已建立應用程式。您會在下一個步驟中啟動應用程式。

啟動應用程式

在本節中，您會透過 StartApplication 動作來啟動應用程式。

啟動應用程式

1. 將下列 JSON 程式碼複製到名為 `start_request.json` 的檔案。

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

2. 以啟動應用程式的上述請求，執行 StartApplication 動作：

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

應用程式現在正在執行。您可以在 Amazon CloudWatch 主控台上查看適用於 Apache Flink 的受管服務指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 StopApplication 動作來停止應用程式。

停止應用程式

1. 將下列 JSON 程式碼複製到名為 `stop_request.json` 的檔案。

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

2. 以停止應用程式的上述請求，執行 StopApplication 動作：

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

現在已停止應用程式。

新增記 CloudWatch 錄選項

您可以使用AWS CLI將 Amazon CloudWatch 日誌串流新增至您的應用程式。如需搭配應用程式使用 CloudWatch 記錄的相關資訊，請參閱the section called “設定日誌”。

更新環境屬性

在本節中，您可以使用 UpdateApplication 動作來變更應用程式的環境屬性，無需重新編譯應用程式的程式碼。在此範例中，您會變更來源和目的地串流的「區域」。

更新應用程式的環境屬性

1. 將下列 JSON 程式碼複製到名為 update_properties_request.json 的檔案。

```
{"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. 使用前述請求執行 UpdateApplication 動作以更新環境屬性：

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

更新應用程式的程式碼

需要使用新版本的程式碼套件更新應用程式的程式碼時，請使用 UpdateApplicationAWS CLI 動作。

> **Note**
>
> 若要載入具有相同檔案名稱的新版應用程式的程式碼，必須指定新的物件版本。如需關於使用 Amazon S3 物件版本的詳細資訊，請參閱啟用或停用版本控制。

若要使用 AWS CLI，請從 Amazon S3 儲存貯體刪除先前的程式碼套件，上傳新版本，然後呼叫 UpdateApplication，指定相同的 Amazon S3 儲存貯體和物件名稱，以及新的物件版本。應用程式將以新的程式碼套件重新啟動。

UpdateApplication 動作的下列範例請求會重新載入應用程式的程式碼並重新啟動應用程式。將 CurrentApplicationVersionId 更新至目前的應用程式版本。您可以使用 ListApplications 或 DescribeApplication 動作來檢查目前的應用程式版本。使用您在the section called “建立兩個 Amazon Kinesis Data Streams”一節中選擇的尾碼更新儲存貯體名稱尾碼 (*<username>*)。

```
{
    "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"
                }
            }
        }
    }
}
```

後續步驟

步驟 4：清除 AWS 資源

步驟 4：清除 AWS 資源

本節包括清除在入門教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

刪除 Managed Service for Apache Flink 應用程式

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis Data Streams

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇，選擇「ExampleOutputStream動作」，選擇「刪除」，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

後續步驟

步驟 5：後續步驟

步驟 5：後續步驟

現在您已建立並執行 Managed Service for Apache Flink 應用程式，請參閱下列資源，取得更進階的 Managed Service for Apache Flink 解決方案。

- Amazon Kinesis 的 AWS 串流資料解決方案：Amazon Kinesis 的 AWS 串流資料解決方案可自動設定輕鬆擷取、存放、處理和交付串流資料所需的 AWS 服務。該解決方案提供了多種解決串流資料使用案例的選項。Apache Flink 的受管理服務選項提供 end-to-end串流 ETL 範例，展示真實世界的應用程式，該應用程式會根據模擬的紐約計程車資料執行分析作業。此解決方案會設定所有必要的 AWS資源，例如 IAM 角色和政策、 CloudWatch 儀表板和 CloudWatch 警示。
- 適用於 Amazon MSK 的 AWS 串流資料解決方案：適用於 Amazon MSK 的 AWS 串流資料解決方案提供 AWS CloudFormation 範本，可讓資料流經生產者、串流儲存、取用者和目的地。

- 使用 Apache Flink 和 Apache Kafka 的點擊流實驗室：點擊流使用案例的端對端實驗室，使用 Amazon Managed Streaming for Apache Kafka 進行串流儲存，使用適用於 Apache Flink 應用程式的 Managed Service for Apache Flink 進行串流處理。
- 適用於 Apache Flink 工作坊的 Amazon 受管服務：在本研討會中，您可以建立 end-to-end 串流架構，以近乎即時的方式擷取、分析串流資料並以視覺化方式呈現。您著手改善紐約市一家出租車公司的運營。您可以近乎即時地分析紐約市計程車車隊的遙測資料，以最佳化其車隊運作。
- Managed Service for Apache Flink：範例：本開發人員指南的這一節提供了在 Managed Service for Apache Flink 中建立和使用應用程式的範例。其中包含範例程式碼和 step-by-step 指示，可協助您為 Apache Flink 應用程式建立受管理服務並測試結果。
- 學習 Flink：動手訓練：官方介紹性 Apache Flink 訓練課程，可協助您開始撰寫可擴展的串流 ETL、分析和事件驅動型應用程式。

> **Note**
>
> 請注意，Managed Service for Apache Flink 不支援本訓練中使用的 Apache Flink 版本 (1.12)。您可以在阿帕奇 Flink 的 Flink 管理服務中使用 Flink 1.15.2。

- 阿帕奇 Flink 代碼示例：一個 GitHub 存儲庫中包含了各種各樣的 Apache Flink 應用實例。

Flink 1.8.2 入門

本主題包含使用 Apache Flink 1.8.2 的 開始使DataStream 用 (API) 教學課程版本。

主題

- Managed Service for Apache Flink 應用程式的元件
- 完成練習的先決條件
- 步驟 1：設定 AWS 帳戶並建立管理員使用者
- 步驟 2：設定 AWS Command Line Interface (AWS CLI)
- 步驟 3：建立並執行 Managed Service for Apache Flink 應用程式
- 步驟 4：清除 AWS 資源

Managed Service for Apache Flink 應用程式的元件

為了處理資料，您的 Managed Service for Apache Flink 應用程式使用 Java/Apache Maven 或 Scala 應用程式來處理輸入，使用 Apache Flink 執行時間生成輸出。

Managed Service for Apache Flink 應用程式包含下列元件：

- 執行時間屬性：您可以使用執行時間屬性來設定應用程式，無需重新編譯應用程式的程式碼。
- 來源：應用程式使用來源來消耗資料。來源連接器從 Kinesis 資料串流、Amazon S3 儲存貯體等讀取資料。如需詳細資訊，請參閱來源。
- 運算子：應用程式會使用一或多個運算子來處理資料。運算子可以轉換、富集或彙總資料。如需詳細資訊，請參閱DataStream API 運算子。
- 接收器：應用程式透過使用接收器生成資料到外部來源。接收器連接器會將資料寫入 Kinesis 資料串流、Kinesis Data Firehose 串流、Amazon S3 儲存貯體等。如需詳細資訊，請參閱接收。

建立、編譯和封裝應用程式的程式碼後，將程式碼套件上傳到 Amazon Simple Storage Service (Amazon S3) 儲存貯體。然後建立 Managed Service for Apache Flink 應用程式。您可以傳入程式碼套件位置，Kinesis 資料串流作為串流資料來源，以及通常是接收應用程式處理後的資料的串流或檔案位置。

完成練習的先決條件

若要完成本指南中的步驟，您必須執行下列各項：

- Java 開發套件 (JDK) 版本 8。將 `JAVA_HOME` 環境變數設為指向您的 JDK 安裝位置。
- 若要在本教學課程中使用 Apache Flink Kinesis 連接器，您必須下載並安裝 Apache Flink。如需詳細資訊，請參閱 將 Apache Flink Kinesis 串流連接器與 Apache Flink 先前版本搭配使用。
- 我們建議您使用開發環境 (如 Eclipse Java Neon 或 IntelliJ Idea) 來開發和編譯您的應用程式。
- Git 用戶端。如果您尚未安裝 Git 用戶端，請先完成安裝。
- Apache Maven 編譯器外掛程式。Maven 必須在您的工作路徑中。若要測試您的 Apache Maven 安裝，輸入以下資訊：

```
$ mvn -version
```

開始執行，請移至 步驟 1：設定 AWS 帳戶並建立管理員使用者。

步驟 1：設定 AWS 帳戶並建立管理員使用者

註冊 AWS 帳戶

如果您還沒有 AWS 帳戶，請完成以下步驟建立新帳戶。

註冊 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   註冊 AWS 帳戶時，會建立 AWS 帳戶根使用者。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為最佳安全實務，將管理存取權指派給管理使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

註冊程序完成後，AWS 會傳送一封確認電子郵件給您。您可以隨時登錄 https://aws.amazon.com/ 並選擇 我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立管理使用者

當您註冊 AWS 帳戶 之後，請保護您的 AWS 帳戶根使用者，啟用 AWS IAM Identity Center，並建立管理使用者，讓您可以不使用根使用者處理日常作業。

保護您的 AWS 帳戶根使用者

1. 選擇 根使用者 並輸入您的 AWS 帳戶電子郵件地址，以帳戶擁有者身分登入 AWS Management Console。在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入使用者指南中的以根使用者身分登入。
2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶根使用者啟用虛擬 MFA 裝置 (主控台)。

建立管理使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，將管理權限授予管理使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的教學課程，請參閱《使用 AWS IAM Identity Center 使用者指南中的以預設 IAM Identity Center 目錄 設定使用者存取權限。

以管理員的身分登入

- 若要使用您的 IAM 身分中心使用者登入，請使用建立 IAM 身分中心使用者時傳送至您電子郵件地址的登入 URL。

  如需有關如何使用 IAM Identity Center 使用者登入的說明，請參閱《AWS 登入 使用者指南》中的登入 AWS存取入口網站。

授予程式設計存取權

若使用者想要與 AWS Management Console 之外的 AWS 互動，則需要程式設計存取權。授予程式設計存取權的方式取決於存取 AWS 的使用者類型。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 設定 AWS CLI 來使用 AWS IAM Identity Center。 • 關於 AWS SDKs、工具和 AWS APIs，請參閱 AWSSDKs 和工具參考指南 中的 IAM Identity Center 驗證。
IAM	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請遵循 IAM 使用者指南 中 使用臨時憑證搭配 AWS 資源 中的指示。
IAM	(不建議使用)	請依照您要使用的介面所提供的指示操作。

哪個使用者需要程式設計存取權？	到	By
	使用長期憑證簽署 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計要求。	• 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 使用 IAM 使用者憑證進行驗證。 • 關於 AWS SDKs 和工具，請參閱 AWSSDKs 和工具參考指南 中的 使用長期憑證進行驗證。 • 關於 AWS API，請參閱 IAM 使用者指南 中的 管理 IAM 使用者的存取金鑰。

步驟 2：設定 AWS Command Line Interface (AWS CLI)

在此步驟中，您將下載並設定與 Amazon Managed Service for Apache Flink 搭配使用的 AWS CLI。

> **Note**
>
> 本指南的入門練習均假設您使用帳戶中的管理員登入資料 (adminuser) 來執行操作。

> **Note**
>
> 如果您已經安裝 AWS CLI，則可能需要升級才能取得最新功能。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的安裝 AWS Command Line Interface。若要查看 AWS CLI 的版本，執行以下命令：
>
> ```
> aws --version
> ```
>
> 此教學課程中的練習需要以下 AWS CLI 版本或更新版本：
>
> ```
> aws-cli/1.16.63
> ```

設定 AWS CLI

1. 下載和設定 AWS CLI。如需說明，請參閱《AWS Command Line Interface 使用者指南》中的下列主題：

   - 安裝 AWS Command Line Interface
   - 設定 AWS CLI

2. 在 AWS CLIconfig 檔案中，為管理員使用者新增具名設定檔。當您執行 AWS CLI 命令時，使用此設定檔。如需具名描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的具名描述檔。

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

   如需可用區域的清單，請參閱 Amazon Web Services 一般參考 中的區域與端點。

   > **Note**
   >
   > 本教學課程中的範例程式碼和命令使用美國西部 (奧勒岡) 區域。若要使用其他 AWS 區域，請將本教學課程的程式碼和指令中的「區域」變更為您要使用的區域。

3. 在命令提示字元中輸入下列 help 命令，以驗證設定：

   ```
   aws help
   ```

設定AWS帳戶之後AWS CLI，您可以嘗試下一個練習，在其中設定範例應用程式並測試 end-to-end 設定。

後續步驟

步驟 3：建立並執行 Managed Service for Apache Flink 應用程式

步驟 3：建立並執行 Managed Service for Apache Flink 應用程式

在本練習中，您會建立 Managed Service for Apache Flink 應用程式，並將資料串流作為來源和目的地。

本節包含下列步驟：

建立兩個 Amazon Kinesis Data Streams

在為本練習建立 Managed Service for Apache Flink 應用程式之前，請先建立兩個 Kinesis 資料串流 (`ExampleInputStream` 和 `ExampleOutputStream`)。您的應用程式會將這些串流用於應用程式來源和目的地串流。

您可以使用 Amazon Kinesis 主控台或以下 AWS CLI 命令來建立這些串流。如需主控台指示，請參閱《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。

建立資料串流 (AWS CLI)

1. 若要建立第一個串流 (`ExampleInputStream`)，請使用以下 Amazon Kinesis `create-stream` AWS CLI 命令。

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

2. 若要建立應用程式用來寫入輸出的第二個串流，請執行相同的命令，將串流名稱變更為 `ExampleOutputStream`。

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

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

1. 使用下列內容建立名為 `stock.py` 的檔案：

```
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. 在教學課程後半段，您會執行 `stock.py` 指令碼來傳送資料至應用程式。

```
$ python stock.py
```

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的 Java 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，執行下列操作：

1. 使用以下指令複製遠端儲存庫：

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

2. 請前往 `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_8` 目錄。

請留意下列與應用程式的程式碼相關的資訊：

- 專案物件模型 (pom.xml) 檔案包含應用程式的組態和相依性資訊，包括 Managed Service for Apache Flink 程式庫。
- `BasicStreamingJob.java` 檔案包含定義應用程式功能的 `main` 方法。
- 應用程式使用 Kinesis 來源從來源串流讀取。以下程式碼片段會建立 Kinesis 來源：

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

- 您的應用程式會建立來源與目的地連接器，以使用 `StreamExecutionEnvironment` 物件來存取外部資源。
- 應用程式會使用靜態屬性來建立來源與目的地連接器。若要使用動態應用程式屬性，請使用 `createSourceFromApplicationProperties` 和 `createSinkFromApplicationProperties` 方法來建立連接器。這些方法會讀取應用程式的屬性，來設定連接器。

  如需執行時間屬性的詳細資訊，請參閱執行時間屬性。

編譯應用程式的程式碼

在本節中，您會使用 Apache Maven 編譯器來建立應用程式的 Java 程式碼。如需安裝 Apache Maven 和 Java 開發套件 (JDK) 的詳細資訊，請參閱 完成練習的先決條件。

> **Note**
>
> 若要將 Kinesis 連接器用於 1.11 之前版本的 Apache Flink，你需要下載、建置和安裝 Apache Maven。如需詳細資訊，請參閱the section called “將 Apache Flink Kinesis 串流連接器與 Apache Flink 先前版本搭配使用”。

編譯應用程式的程式碼

1. 請將應用程式的程式碼編譯並封裝成 JAR 檔案，以使用應用程式的程式碼。您可以使用下列兩種方式的其中之一，編譯和封裝您的程式碼：

   - 使用命令列 Maven 工具。請在包含 `pom.xml` 檔案的目錄中執行下列命令，來建立 JAR 檔案：

     ```
     mvn package -Dflink.version=1.8.2
     ```

   - 設定開發環境。如需詳細資訊，請參閱您的開發環境文件。

     > **Note**
     >
     > 提供的來源程式碼依賴於 Java 1.8 中的程式庫。確保您專案的 Java 版本是 1.8。

   您可以將您的套件做為 JAR 檔案上傳，或壓縮您的套件並做為 ZIP 檔案上傳。如果是使用 AWS CLI 建立應用程式，您會指定程式碼內容類型 (JAR 或 ZIP)。

2. 如果編譯時發生錯誤，請確認您的 `JAVA_HOME` 環境變數是否正確設定。

如果應用程式成功編譯，則會建立下列檔案：

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

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會建立 Amazon Simple Storage Service (Amazon S3) 儲存貯體並上傳您的應用程式的程式碼。

上傳應用程式的程式碼

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 選擇建立儲存貯體。
3. 在儲存貯體名稱欄位中，輸入 **ka-app-code-*<username>***。新增尾碼至儲存貯體名稱，例如您的使用者名稱，使其成為全域唯一的。選擇下一步。
4. 在設定選項步驟中，保留原有設定並選擇 Next (下一步)。
5. 在設定許可步驟中，保留原有設定並選擇 Next (下一步)。
6. 選擇建立儲存貯體。
7. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
8. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `aws-kinesis-analytics-java-apps-1.0.jar` 檔案。選擇下一步。
9. 您不需要變更物件的任何設定，因此請選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

您可以使用主控台或 AWS CLI 建立和執行 Managed Service for Apache Flink 應用程式。

> **Note**
>
> 使用主控台建立應用程式時，系統會為您建立 AWS Identity and Access Management (IAM) 和 Amazon CloudWatch 日誌資源。當您使用 AWS CLI 建立應用程式時，則會個別建立這些資源。

主題

- 建立和執行應用程式 (主控台)
- 建立並執行應用程式 (AWS CLI)

建立和執行應用程式 (主控台)

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 對於 Description (說明)，輸入 **My java test app**。
   - 針對執行時間，選擇 Apache Flink。
   - 將版本下拉式清單保留為 Apache Flink 1.8 版 (建議版本)。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：kinesis-analytics-service-*MyApplication*-*us-west-2*
> - 角色：kinesisanalytics-*MyApplication*-*us-west-2*

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **aws-kinesis-analytics-java-apps-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 輸入以下應用程式屬性和數值：

群組 ID	金鑰	值
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**

群組 ID	金鑰	值
ProducerConfigProperties	**AggregationEnabled**	**false**

5. 在監控下，確保監控指標層級設為應用程式。
6. 若要CloudWatch 記錄，請選取 [啟用] 核取方塊。
7. 選擇更新。

> **Note**
>
> 當您選擇啟用 Amazon 日誌 CloudWatch 記錄時，Apache Flink 的受管服務會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`

執行應用程式

1. 在MyApplication頁面上，選擇 [執行]。確認動作。
2. 應用程式執行時，重新整理頁面。主控台會顯示 Application graph (應用程式圖形)。

停止應用程式

在MyApplication頁面上，選擇 [停止]。確認動作。

更新應用程式

您可以使用主控台更新應用程式設定，例如應用程式屬性、監控設定及位置或應用程式 JAR 的檔名。如果需要更新應用程式的程式碼，也可以從 Amazon S3 儲存貯體重新載入應用程式 JAR。

在MyApplication頁面上，選擇設定。更新應用程式設定，然後選擇更新。

建立並執行應用程式 (AWS CLI)

在本節中，您可以使用 AWS CLI 建立和執行 Managed Service for Apache Flink 應用程式。Managed Service for Apache Flink 使用 `kinesisanalyticsv2` AWS CLI 命令建立 Managed Service for Apache Flink 應用程式並與之互動。

建立許可政策

> **Note**
>
> 您必須為應用程式建立許可政策和角色。如果您未建立這些 IAM 資源，應用程式將無法存取其資料和日誌串流。

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 `read` 動作的許可，而另一條則是授與目的地串流上 `write` 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

使用以下程式碼來建立 `AKReadSourceStreamWriteSinkStream` 許可政策。以您用於建立 Amazon S3 儲存貯體 (以儲存應用程式的程式碼) 的使用者名稱來取代 *`username`*。以您的帳戶 ID 取代 Amazon Resource Names (ARNs) (*`012345678901`*) 中的帳戶 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"
        }
    ]
}
```

如需建立許可政策的指 step-by-step 示，請參閱 IAM 使用指南中的教學課程：建立和附加您的第一個客戶受管政策。

> **Note**
>
> 若要存取其他 Amazon 服務，您可以使用 AWS SDK for Java。Managed Service for Apache Flink 自動將 SDK 所需的憑證設定為與應用程式相關聯的服務執行 IAM 角色。無須採取額外的步驟。

建立 IAM 角色

在本節中，您會建立 Managed Service for Apache Flink 應用程式可以擔任的 IAM 角色，以便讀取來源串流與寫入目的地串流。

Managed Service for Apache Flink 沒有許可，無法存取串流。您可以透過 IAM 角色來授與這些許可。各 IAM 角色都有連接兩項政策。信任政策會授與擔任角色的 Managed Service for Apache Flink 許可，而許可政策決定了 Managed Service for Apache Flink 在擔任角色後可以執行的作業。

您會將在上一節中建立的許可政策連接至此角色。

若要建立一個 IAM 角色

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Roles (角色) 、Create Role (建立角色)。
3. 在選取可信身分類型下，選擇 AWS 服務。在 Choose the service that will use this role (選擇將使用此角色的服務) 下，選擇 Kinesis。在 Select your use case (選取您的使用案例) 下，選擇 Kinesis Analytics (Kinesis 分析)。

   選擇 Next: Permissions (下一步：許可)。
4. 在 Attach permissions policies (連接許可政策) 頁面上，選擇 Next: Review (下一步：檢閱)。您會在建立角色後連接許可政策。
5. 在建立角色頁面，輸入 **MF-stream-rw-role** 作為角色名稱。選擇建立角色。

現在您已建立新的 IAM 角色，名為 `MF-stream-rw-role`。您接著會更新角色的信任和許可政策。

6. 將 許可政策連接到角色。

> **Note**
>
> 在此練習中，Managed Service for Apache Flink 擔任從 Kinesis 資料串流 (來源) 讀取資料並將輸出寫入另一個 Kinesis 資料串流的角色。因此您會連接在上一個步驟中建立的政策，the section called “建立許可政策”。

   a. 在摘要頁面上，選擇許可標籤。

   b. 選擇 Attach Policies (連接政策)。

   c. 在搜尋方塊中，輸入 **AKReadSourceStreamWriteSinkStream** (您在上一節中建立的政策)。

   d. 選擇 AK 原ReadSourceStreamWriteSinkStream則，然後選擇 [附加原則]。

您現在已建立應用程式用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的指 step-by-step 示，請參閱 IAM 使用者指南中的建立 IAM 角色 (主控台)。

建立 Managed Service for Apache Flink 應用程式

1. 將下列 JSON 程式碼複製到名為 `create_request.json` 的檔案。以您之前建立之角色的 ARN，取代範例角色 ARN。以您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (*username*)。以您的帳戶 ID 取代服務執行角色中的範例帳戶 ID (*012345678901*)。

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_8",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-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. 以建立應用程式的上述請求，執行 CreateApplication 動作：

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

現在已建立應用程式。您會在下一個步驟中啟動應用程式。

啟動應用程式

在本節中，您會透過 StartApplication 動作來啟動應用程式。

啟動應用程式

1. 將下列 JSON 程式碼複製到名為 start_request.json 的檔案。

```
{
```

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

2. 以啟動應用程式的上述請求，執行 StartApplication 動作：

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

應用程式現在正在執行。您可以在 Amazon CloudWatch 主控台上查看適用於 Apache Flink 的受管服務指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 StopApplication 動作來停止應用程式。

停止應用程式

1. 將下列 JSON 程式碼複製到名為 stop_request.json 的檔案。

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

2. 以停止應用程式的上述請求，執行 StopApplication 動作：

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

現在已停止應用程式。

新增記 CloudWatch 錄選項

您可以使用AWS CLI將 Amazon CloudWatch 日誌串流新增至您的應用程式。如需搭配應用程式使用 CloudWatch 記錄的相關資訊，請參閱the section called “設定日誌”。

更新環境屬性

在本節中，您可以使用 UpdateApplication 動作來變更應用程式的環境屬性，無需重新編譯應用程式的程式碼。在此範例中，您會變更來源和目的地串流的「區域」。

更新應用程式的環境屬性

1. 將下列 JSON 程式碼複製到名為 update_properties_request.json 的檔案。

```
{"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. 使用前述請求執行 UpdateApplication 動作以更新環境屬性：

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

更新應用程式的程式碼

需要使用新版本的程式碼套件更新應用程式的程式碼時，請使用 UpdateApplicationAWS CLI 動作。

> **Note**
>
> 若要載入具有相同檔案名稱的新版應用程式的程式碼，必須指定新的物件版本。如需關於使用 Amazon S3 物件版本的詳細資訊，請參閱啟用或停用版本控制。

若要使用 AWS CLI，請從 Amazon S3 儲存貯體刪除先前的程式碼套件，上傳新版本，然後呼叫 `UpdateApplication`，指定相同的 Amazon S3 儲存貯體和物件名稱，以及新的物件版本。應用程式將以新的程式碼套件重新啟動。

`UpdateApplication` 動作的下列範例請求會重新載入應用程式的程式碼並重新啟動應用程式。將 `CurrentApplicationVersionId` 更新至目前的應用程式版本。您可以使用 `ListApplications` 或 `DescribeApplication` 動作來檢查目前的應用程式版本。使用您在the section called “建立兩個 Amazon Kinesis Data Streams”一節中選擇的尾碼更新儲存貯體名稱尾碼 (*<username>*)。

```
{
    "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"
                }
            }
        }
    }
}
```

後續步驟

步驟 4：清除 AWS 資源

步驟 4：清除 AWS 資源

本節包括清除在入門教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis Data Streams
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 選擇設定。
4. 在快照區段中，選擇停用，然後選擇更新。
5. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis Data Streams

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇，選擇「ExampleOutputStream動作」，選擇「刪除」，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。

3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

Flink 1.6.2 入門

本主題包含使用 Apache Flink 1.6.2 的 開始使DataStream 用 (API) 教學課程版本。

主題

- Managed Service for Apache Flink 的元件
- 完成練習的先決條件
- 步驟 1：設定 AWS 帳戶並建立管理員使用者
- 步驟 2：設定 AWS Command Line Interface (AWS CLI)
- 步驟 3：建立並執行 Managed Service for Apache Flink 應用程式
- 步驟 4：清除 AWS 資源

Managed Service for Apache Flink 的元件

為了處理資料，您的 Managed Service for Apache Flink 應用程式使用 Java/Apache Maven 或 Scala 應用程式來處理輸入，使用 Apache Flink 執行時間生成輸出。

Managed Service for Apache Flink 應用程式包含以下元件：

- 執行時間屬性：您可以使用執行時間屬性來設定應用程式，無需重新編譯應用程式的程式碼。
- 來源：應用程式使用來源來消耗資料。來源連接器從 Kinesis 資料串流、Amazon S3 儲存貯體等讀取資料。如需詳細資訊，請參閱來源。
- 運算子：應用程式會使用一或多個運算子來處理資料。運算子可以轉換、富集或彙總資料。如需詳細資訊，請參閱DataStream API 運算子。
- 接收器：應用程式透過使用接收器生成資料到外部來源。接收器連接器會將資料寫入 Kinesis 資料串流、Kinesis Data Firehose 串流、Amazon S3 儲存貯體等。如需詳細資訊，請參閱 接收。

建立、編譯和封裝應用程式後，將程式碼套件上傳到 Amazon Simple Storage Service (Amazon S3) 儲存貯體。然後建立 Managed Service for Apache Flink 應用程式。您可以傳入程式碼套件位置，Kinesis 資料串流作為串流資料來源，以及通常是接收應用程式處理後的資料的串流或檔案位置。

完成練習的先決條件

若要完成本指南中的步驟，您必須執行下列各項：

- Java 開發套件 (JDK) 版本 8。將 JAVA_HOME 環境變數設為指向您的 JDK 安裝位置。
- 我們建議您使用開發環境 (如 Eclipse Java Neon 或 IntelliJ Idea) 來開發和編譯您的應用程式。
- Git 用戶端。 如果您尚未安裝 Git 用戶端，請先完成安裝。
- Apache Maven 編譯器外掛程式。Maven 必須在您的工作路徑中。若要測試您的 Apache Maven 安裝，輸入以下資訊：

```
$ mvn -version
```

開始執行，請移至 步驟 1：設定 AWS 帳戶並建立管理員使用者。

步驟 1：設定 AWS 帳戶並建立管理員使用者

註冊 AWS 帳戶

如果您還沒有 AWS 帳戶，請完成以下步驟建立新帳戶。

註冊 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

註冊 AWS 帳戶時，會建立 AWS 帳戶根使用者。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為最佳安全實務，將管理存取權指派給管理使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

註冊程序完成後，AWS 會傳送一封確認電子郵件給您。您可以隨時登錄 https://aws.amazon.com/ 並選擇 我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立管理使用者

當您註冊 AWS 帳戶 之後，請保護您的 AWS 帳戶根使用者，啟用 AWS IAM Identity Center，並建立管理使用者，讓您可以不使用根使用者處理日常作業。

保護您的 AWS 帳戶根使用者

1. 選擇 根使用者 並輸入您的 AWS 帳戶電子郵件地址，以帳戶擁有者身分登入 AWS Management Console。在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入使用者指南中的以根使用者身分登入。
2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶根使用者啟用虛擬 MFA 裝置 (主控台)。

建立管理使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，將管理權限授予管理使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的教學課程，請參閱《使用 AWS IAM Identity Center 使用者指南中的以預設 IAM Identity Center 目錄 設定使用者存取權限。

以管理員的身分登入

- 若要使用您的 IAM 身分中心使用者登入，請使用建立 IAM 身分中心使用者時傳送至您電子郵件地址的登入 URL。

如需有關如何使用 IAM Identity Center 使用者登入的說明，請參閱《AWS 登入 使用者指南》中的登入 AWS存取入口網站。

授予程式設計存取權

若使用者想要與 AWS Management Console 之外的 AWS 互動，則需要程式設計存取權。授予程式設計存取權的方式取決於存取 AWS 的使用者類型。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 設定 AWS CLI 來使用 AWS IAM Identity Center。 • 關於 AWS SDKs、工具和 AWS APIs，請參閱 AWSSDKs 和工具參考指南 中的 IAM Identity Center 驗證。
IAM	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請遵循 IAM 使用者指南 中 使用臨時憑證搭配 AWS 資源 中的指示。
IAM	(不建議使用) 使用長期憑證簽署 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 使

哪個使用者需要程式設計存取權？	到	By
		用 IAM 使用者憑證進行驗證。 • 關於 AWS SDKs 和工具，請參閱 AWSSDKs 和工具參考指南 中的 使用長期憑證進行驗證。 • 關於 AWS API，請參閱 IAM 使用者指南 中的 管理 IAM 使用者的存取金鑰。

步驟 2：設定 AWS Command Line Interface (AWS CLI)

在此步驟中，您將下載並設定與 Amazon Managed Service for Apache Flink 搭配使用的 AWS CLI。

> **Note**
>
> 本指南的入門練習均假設您使用帳戶中的管理員登入資料 (adminuser) 來執行操作。

> **Note**
>
> 如果您已經安裝 AWS CLI，則可能需要升級才能取得最新功能。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的安裝 AWS Command Line Interface。若要查看 AWS CLI 的版本，執行以下命令：
>
> ```
> aws --version
> ```
>
> 此教學課程中的練習需要以下 AWS CLI 版本或更新版本：
>
> ```
> aws-cli/1.16.63
> ```

設定 AWS CLI

1. 下載和設定 AWS CLI。如需說明，請參閱《AWS Command Line Interface 使用者指南》中的下列主題：

   - 安裝 AWS Command Line Interface
   - 設定 AWS CLI

2. 在 AWS CLIconfig 檔案中，為管理員使用者新增具名設定檔。當您執行 AWS CLI 命令時，使用此設定檔。如需具名描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的具名描述檔。

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

   如需可用 AWS 區域的清單，請參閱 Amazon Web Services 一般參考 中的 區域與端點。

   > **Note**
   >
   > 本教學課程中的範例程式碼和命令使用美國西部 (奧勒岡) 區域。若要使用其他區域，請將本教學課程的程式碼和指令中的「區域」變更為您要使用的區域。

3. 在命令提示字元中輸入下列 help 命令，以驗證設定：

   ```
   aws help
   ```

設定AWS帳戶之後AWS CLI，您可以嘗試下一個練習，在其中設定範例應用程式並測試 end-to-end 設定。

後續步驟

步驟 3：建立並執行 Managed Service for Apache Flink 應用程式

步驟 3：建立並執行 Managed Service for Apache Flink 應用程式

在本練習中，您會建立 Managed Service for Apache Flink 應用程式，並將資料串流作為來源和目的地。

本節包含下列步驟：

- 建立兩個 Amazon Kinesis Data Streams
- 寫入範例記錄至輸入串流
- 下載並檢查 Apache Flink 串流 Java 程式碼
- 編譯應用程式的程式碼
- 上傳 Apache Flink 串流 Java 程式碼
- 建立並執行 Managed Service for Apache Flink 應用程式

建立兩個 Amazon Kinesis Data Streams

在為本練習建立 Managed Service for Apache Flink 應用程式之前，請先建立兩個 Kinesis 資料串流 (`ExampleInputStream` 和 `ExampleOutputStream`)。您的應用程式會將這些串流用於應用程式來源和目的地串流。

您可以使用 Amazon Kinesis 主控台或以下 AWS CLI 命令來建立這些串流。如需主控台指示，請參閱《Amazon Kinesis Data Streams 開發人員指南》中的建立和更新資料串流。

建立資料串流 (AWS CLI)

1. 若要建立第一個串流 (`ExampleInputStream`)，請使用以下 Amazon Kinesis `create-stream` AWS CLI 命令。

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

2. 若要建立應用程式用來寫入輸出的第二個串流，請執行相同的命令，將串流名稱變更為 `ExampleOutputStream`。

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

寫入範例記錄至輸入串流

在本節，您會使用 Python 指令碼將範例記錄寫入供應用程式處理的串流。

> **Note**
>
> 本節需要 AWS SDK for Python (Boto)。

1. 使用下列內容建立名為 `stock.py` 的檔案：

```
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. 在教學課程後半段，您會執行 `stock.py` 指令碼來傳送資料至應用程式。

```
$ python stock.py
```

下載並檢查 Apache Flink 串流 Java 程式碼

此範例的 Java 應用程式程式碼可從中取得 GitHub。若要下載應用程式的程式碼，執行下列操作：

1. 使用以下指令複製遠端儲存庫：

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

2. 請前往 `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_6` 目錄。

請留意下列與應用程式的程式碼相關的資訊：

- 專案物件模型 (pom.xml) 檔案包含應用程式的組態和相依性資訊，包括 Managed Service for Apache Flink 程式庫。
- `BasicStreamingJob.java` 檔案包含定義應用程式功能的 `main` 方法。
- 應用程式使用 Kinesis 來源從來源串流讀取。以下程式碼片段會建立 Kinesis 來源：

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

- 您的應用程式會建立來源與目的地連接器，以使用 `StreamExecutionEnvironment` 物件來存取外部資源。
- 應用程式會使用靜態屬性來建立來源與目的地連接器。若要使用動態應用程式屬性，請使用 `createSourceFromApplicationProperties` 和 `createSinkFromApplicationProperties` 方法來建立連接器。這些方法會讀取應用程式的屬性，來設定連接器。

  如需執行時間屬性的詳細資訊，請參閱執行時間屬性。

編譯應用程式的程式碼

在本節中，您會使用 Apache Maven 編譯器來建立應用程式的 Java 程式碼。如需安裝 Apache Maven 和 Java 開發套件 (JDK) 的詳細資訊，請參閱 完成練習的先決條件。

> **Note**
>
> 若要將 Kinesis 連接器用於 1.11 之前版本的 Apache Flink，您需要下載連接器的來源程式碼，並如《Apache Flink 文件》https://ci.apache.org/projects/flink/flink-docs-release-1.6/dev/connectors/kinesis.html所述建置。

編譯應用程式的程式碼

1. 請將應用程式的程式碼編譯並封裝成 JAR 檔案，以使用應用程式的程式碼。您可以使用下列兩種方式的其中之一，編譯和封裝您的程式碼：

   - 使用命令列 Maven 工具。請在包含 `pom.xml` 檔案的目錄中執行下列命令，來建立 JAR 檔案：

     ```
     mvn package
     ```

     > **Note**
     >
     > Managed Service for Apache Flink Runtime 1.0.1 版本不需要 -Dflink.version 參數；只有版本 1.1.0 及更新版本才需要此參數。如需詳細資訊，請參閱 the section called “指定應用程式的 Apache Flink 版本”。

   - 設定開發環境。如需詳細資訊，請參閱您的開發環境文件。

   您可以將您的套件做為 JAR 檔案上傳，或壓縮您的套件並做為 ZIP 檔案上傳。如果是使用 AWS CLI 建立應用程式，您會指定程式碼內容類型 (JAR 或 ZIP)。
2. 如果編譯時發生錯誤，請確認您的 `JAVA_HOME` 環境變數是否正確設定。

如果應用程式成功編譯，則會建立下列檔案：

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

上傳 Apache Flink 串流 Java 程式碼

在本節中，您會建立 Amazon Simple Storage Service (Amazon S3) 儲存貯體並上傳您的應用程式的程式碼。

上傳應用程式的程式碼

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 選擇建立儲存貯體。
3. 在儲存貯體名稱欄位中，輸入 **ka-app-code-*<username>***。新增尾碼至儲存貯體名稱，例如您的使用者名稱，使其成為全域唯一的。選擇下一步。
4. 在設定選項步驟中，保留原有設定並選擇 Next (下一步)。
5. 在設定許可步驟中，保留原有設定並選擇 Next (下一步)。
6. 選擇建立儲存貯體。
7. 在 Amazon S3 主控台中，選擇 ka-app-code- 儲存<username>貯體，然後選擇「上傳」。
8. 在選取檔案步驟中，選擇 新增檔案。導覽至您在上一步驟中建立的 `aws-kinesis-analytics-java-apps-1.0.jar` 檔案。選擇下一步。
9. 在設定許可步驟中，保留原有設定。選擇下一步。
10. 在設定屬性步驟中，保留原有設定。選擇上傳。

您的應用程式的程式碼現在儲存在您的應用程式可以存取的 Amazon S3 儲存貯體中。

建立並執行 Managed Service for Apache Flink 應用程式

您可以使用主控台或 AWS CLI 建立和執行 Managed Service for Apache Flink 應用程式。

> **Note**
>
> 使用主控台建立應用程式時，系統會為您建立 AWS Identity and Access Management (IAM) 和 Amazon CloudWatch 日誌資源。當您使用 AWS CLI 建立應用程式時，則會個別建立這些資源。

主題

- 建立和執行應用程式 (主控台)
- 建立並執行應用程式 (AWS CLI)

建立和執行應用程式 (主控台)

依照這些步驟來使用主控台建立、設定、更新及執行應用程式。

建立應用程式

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在 Managed Service for Apache Flink 儀表板上，選擇建立分析應用程式。
3. 在 Managed Service for Apache Flink - 建立應用程式頁面，提供應用程式詳細資訊，如下所示：
   - 在應用程式名稱中，輸入 **MyApplication**。
   - 對於 Description (說明)，輸入 **My java test app**。
   - 針對執行時間，選擇 Apache Flink。

     > **Note**
     >
     > Managed Service for Apache Flink 使用 Apache Flink 1.8.2 或 1.6.2 版。

   - 將版本下拉式清單變更為 Apache Flink 1.6。
4. 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
5. 選擇 建立應用程式。

> **Note**
>
> 使用主控台建立 Managed Service for Apache Flink 應用程式時，可以選擇是否為應用程式建立 IAM 角色和政策。應用程式使用此角色和政策來存取其相依資源。這些 IAM 資源會如下所述使用您的應用程式名稱和區域命名：
>
> - 政策：`kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - 角色：`kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

編輯 IAM 政策

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 選擇政策。選擇主控台為您在上一節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。

4. 將下列政策範例的反白部分新增至政策。以您的帳戶 ID 取代範例帳戶 ID (*012345678901*)。

```
{
    "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"
        }
    ]
}
```

設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在設定應用程式頁面，提供程式碼位置：
   - 對於 Amazon S3 儲存貯體，請輸入 **ka-app-code-*<username>***。
   - 對於 Amazon S3 物件的路徑，請輸入 **java-getting-started-1.0.jar**。
3. 在存取應用程式資源下，對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
4. 輸入以下應用程式屬性和數值：

群組 ID	金鑰	值
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**

群組 ID	金鑰	值
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

5. 在監控下，確保監控指標層級設為應用程式。
6. 若要CloudWatch 記錄，請選取 [啟用] 核取方塊。
7. 選擇更新。

> **Note**
>
> 當您選擇啟用 Amazon 日誌 CloudWatch 記錄時，Apache Flink 的受管服務會為您建立日誌群組和日誌串流。這些資源的名稱如下所示：
>
> - 日誌群組：`/aws/kinesis-analytics/MyApplication`
> - 日誌串流：`kinesis-analytics-log-stream`

執行應用程式

1. 在MyApplication頁面上，選擇 [執行]。確認動作。
2. 應用程式執行時，重新整理頁面。主控台會顯示 Application graph (應用程式圖形)。

停止應用程式

在MyApplication頁面上，選擇 [停止]。確認動作。

更新應用程式

您可以使用主控台更新應用程式設定，例如應用程式屬性、監控設定及位置或應用程式 JAR 的檔名。如果需要更新應用程式的程式碼，也可以從 Amazon S3 儲存貯體重新載入應用程式 JAR。

在MyApplication頁面上，選擇設定。更新應用程式設定，然後選擇更新。

建立並執行應用程式 (AWS CLI)

在本節中，您可以使用 AWS CLI 建立和執行 Managed Service for Apache Flink 應用程式。Managed Service for Apache Flink 使用 `kinesisanalyticsv2` AWS CLI 命令建立 Managed Service for Apache Flink 應用程式並與之互動。

建立許可政策

您會先建立具有兩條陳述式的許可政策：一條陳述式授與來源串流上 `read` 動作的許可，而另一條則是授與目的地串流上 `write` 動作的許可。您之後會將政策連接至 IAM 角色 (您會在下一節中建立)。因此，當 Managed Service for Apache Flink 擔任角色時，服務便具有從來源串流讀取並寫入目的地串流的所需許可。

使用以下程式碼來建立 `AKReadSourceStreamWriteSinkStream` 許可政策。以您用於建立 Amazon S3 儲存貯體 (以儲存應用程式的程式碼) 的使用者名稱來取代 *`username`*。以您的帳戶 ID 取代 Amazon Resource Names (ARNs) (*`012345678901`*) 中的帳戶 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"
        }
    ]
}
```

如需建立許可政策的指 step-by-step 示，請參閱 IAM 使用指南中的教學課程：建立和附加您的第一個客戶受管政策。

> **Note**
>
> 若要存取其他 Amazon 服務，您可以使用 AWS SDK for Java。Managed Service for Apache Flink 自動將 SDK 所需的憑證設定為與應用程式相關聯的服務執行 IAM 角色。無須採取額外的步驟。

建立 IAM 角色

在本節中，您會建立 Managed Service for Apache Flink 應用程式可以擔任的 IAM 角色，以便讀取來源串流與寫入目的地串流。

Managed Service for Apache Flink 沒有許可，無法存取串流。您可以透過 IAM 角色來授與這些許可。各 IAM 角色都有連接兩項政策。信任政策會授與擔任角色的 Managed Service for Apache Flink 許可，而許可政策決定了 Managed Service for Apache Flink 在擔任角色後可以執行的作業。

您會將在上一節中建立的許可政策連接至此角色。

若要建立一個 IAM 角色

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Roles (角色) 、Create Role (建立角色)。
3. 在選取可信身分類型下，選擇 AWS 服務。在 Choose the service that will use this role (選擇將使用此角色的服務) 下，選擇 Kinesis。在 Select your use case (選取您的使用案例) 下，選擇 Kinesis Analytics (Kinesis 分析)。

   選擇 Next: Permissions (下一步：許可)。
4. 在 Attach permissions policies (連接許可政策) 頁面上，選擇 Next: Review (下一步：檢閱)。您會在建立角色後連接許可政策。
5. 在建立角色頁面，輸入 **MF-stream-rw-role** 作為角色名稱。選擇建立角色。

現在您已建立新的 IAM 角色，名為 `MF-stream-rw-role`。您接著會更新角色的信任和許可政策。

6. 將 許可政策連接到角色。

> **Note**
>
> 在此練習中，Managed Service for Apache Flink 擔任從 Kinesis 資料串流 (來源) 讀取資料並將輸出寫入另一個 Kinesis 資料串流的角色。因此您會連接在上一個步驟中建立的政策，the section called “建立許可政策”。

   a. 在摘要頁面上，選擇許可標籤。

   b. 選擇 Attach Policies (連接政策)。

   c. 在搜尋方塊中，輸入 **AKReadSourceStreamWriteSinkStream** (您在上一節中建立的政策)。

   d. 選擇 AK 原ReadSourceStreamWriteSinkStream則，然後選擇 [附加原則]。

您現在已建立應用程式用於存取資源的服務執行角色。請記下新角色的 ARN。

如需建立角色的指 step-by-step 示，請參閱 IAM 使用者指南中的建立 IAM 角色 (主控台)。

建立 Managed Service for Apache Flink 應用程式

1. 將下列 JSON 程式碼複製到名為 `create_request.json` 的檔案。以您之前建立之角色的 ARN，取代範例角色 ARN。以您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (*username*)。以您的帳戶 ID 取代服務執行角色中的範例帳戶 ID (*012345678901*)。

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_6",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-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. 以建立應用程式的上述請求，執行 CreateApplication 動作：

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

現在已建立應用程式。您會在下一個步驟中啟動應用程式。

啟動應用程式

在本節中，您會透過 StartApplication 動作來啟動應用程式。

啟動應用程式

1. 將下列 JSON 程式碼複製到名為 start_request.json 的檔案。

```
{
```

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

2. 以啟動應用程式的上述請求，執行 StartApplication 動作：

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

應用程式現在正在執行。您可以在 Amazon CloudWatch 主控台上查看適用於 Apache Flink 的受管服務指標，以確認應用程式是否正常運作。

停止應用程式

在本節，您會使用該 StopApplication 動作來停止應用程式。

停止應用程式

1. 將下列 JSON 程式碼複製到名為 stop_request.json 的檔案。

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

2. 以停止應用程式的上述請求，執行 StopApplication 動作：

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

現在已停止應用程式。

新增記 CloudWatch 錄選項

您可以使用AWS CLI將 Amazon CloudWatch 日誌串流新增至您的應用程式。如需搭配應用程式使用 CloudWatch 記錄的相關資訊，請參閱the section called “設定日誌”。

更新環境屬性

在本節中，您可以使用 UpdateApplication 動作來變更應用程式的環境屬性，無需重新編譯應用程式的程式碼。在此範例中，您會變更來源和目的地串流的「區域」。

更新應用程式的環境屬性

1. 將下列 JSON 程式碼複製到名為 update_properties_request.json 的檔案。

```
{"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. 使用前述請求執行 UpdateApplication 動作以更新環境屬性：

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

更新應用程式的程式碼

需要使用新版本的程式碼套件更新應用程式的程式碼時，請使用 UpdateApplicationAWS CLI 動作。

若要使用 AWS CLI，請從 Amazon S3 儲存貯體刪除先前的程式碼套件，上傳新版本，然後呼叫 `UpdateApplication`，指定相同的 Amazon S3 儲存貯體和物件名稱。應用程式將以新的程式碼套件重新啟動。

`UpdateApplication` 動作的下列範例請求會重新載入應用程式的程式碼並重新啟動應用程式。將 `CurrentApplicationVersionId` 更新至目前的應用程式版本。您可以使用 `ListApplications` 或 `DescribeApplication` 動作來檢查目前的應用程式版本。使用您在the section called “建立兩個 Amazon Kinesis Data Streams”一節中選擇的尾碼更新儲存貯體名稱尾碼 (*<username>*)。

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

步驟 4：清除 AWS 資源

本節包括清除在入門教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- 刪除 Managed Service for Apache Flink 應用程式
- 刪除 Kinesis Data Streams
- 刪除 Amazon S3 物件與儲存貯體
- 刪除 IAM 資源
- 刪除您的 CloudWatch 資源

刪除 Managed Service for Apache Flink 應用程式

1. 在以下網址開啟 Kinesis 主控台：https://console.aws.amazon.com/kinesis。

2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 選擇設定。
4. 在快照區段中，選擇停用，然後選擇更新。
5. 在應用程式的頁面中，選擇刪除，然後確認刪除。

刪除 Kinesis Data Streams

1. 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
2. 在「Kinesis Data Streams」面板中，選擇ExampleInputStream。
3. 在ExampleInputStream頁面中，選擇「刪除 Kinesis 串流」，然後確認刪除。
4. 在 Kinesis 串流頁面中，選擇，選擇「ExampleOutputStream動作」，選擇「刪除」，然後確認刪除。

刪除 Amazon S3 物件與儲存貯體

1. 在以下網址開啟 Amazon S3 主控台：https://console.aws.amazon.com/s3/。
2. 選擇 ka-app-code- 桶。 <username>
3. 選擇刪除，然後輸入儲存貯體名稱以確認刪除。

刪除 IAM 資源

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽列中，選擇政策。
3. 在篩選器控制項中，輸入 kinesis。
4. 選擇 kinesis-analytics-service--MyApplication 美西 -2 原則。
5. 選擇政策動作，然後選擇刪除。
6. 在導覽列中，選擇角色。
7. 選擇運動分析-MyApplication 美西 -2 角色。
8. 選擇刪除角色，然後確認刪除。

刪除您的 CloudWatch 資源

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

2. 在導覽窗格中，選擇日誌。
3. 選擇 /aws/運動分析/日誌群MyApplication組。
4. 選擇刪除日誌群組，然後確認刪除。

Apache Flink 設定

Managed Service for Apache Flink 是 Apache Flink 框架的實作。Managed Service for Apache Flink 使用本節所述的預設值。其中一些值可由 Managed Service for Apache Flink 應用程式在程式碼中設定，其他值則無法變更。

本主題包含下列章節：

- Apache Flink 組態
- 狀態後端
- 檢查點
- 儲存點
- 堆積大小
- 緩衝區消脹
- 可修改的 Flink 組態屬性
- 檢視已設定的 Flink 屬性

Apache Flink 組態

Managed Service for Apache Flink 提供預設的 Flink 組態，其中包含大多數屬性的 Apache Flink 建議值，少數一些基於常用應用程式設定檔。如需 Flink 組態的詳細資訊，請參閱組態。服務提供的預設組態適用於大多數應用程式。不過，如果您需要調整 Flink 組態屬性，以改善具有高平行處理層級、高記憶體和狀態使用率的某些應用程式之效能，或在 Apache Flink 中啟用新的偵錯功能，您可以透過請求支援案例來變更某些屬性。如需詳細資訊，請參閱 AWS 支援中心。您可以使用 Apache Flink 儀表板檢查應用程式的目前組態。

狀態後端

Managed Service for Apache Flink 將暫時性資料儲存在狀態後端。Managed Service for Apache Flink 使用 RocksDBStateBackend。呼叫 `setStateBackend` 來設定不同的後端沒有任何效果。

我們在狀態後端啟用以下功能：

- 增量狀態後端快照

- 非同步狀態後端快照
- 檢查點本機復原

在 Managed Service for Apache Flink 中，`state.backend.rocksdb.ttl.compaction.filter.enabled` 組態預設為啟用。使用此篩選器，您可以更新應用程式的程式碼，以啟用壓縮清理策略。如需詳細資訊，請參閱https://nightlies.apache.org/flink/flink-docs-release-1.15/《Apche Flink 文件》中的 Flink 1.8.0 中的狀態 TTL。

如需狀態後端的詳細資訊，請參閱《Apache F link 文件》中的狀態後端。https://nightlies.apache.org/flink/flink-docs-release-1.15/

檢查點

Managed Service for Apache Flink 使用具有下列值的預設檢查點組態。其中一些值可以變更。必須將 CheckpointConfiguration.ConfigurationType 設定為 `CUSTOM`，Managed Service for Apache Flink 才能使用修改的檢查點值。

設定	可以修改嗎？	方法	預設值
CheckpointingEnabled	可修改	建立應用程式 更新應用程式 AWS CloudFormation	True
CheckpointInterval	可修改	建立應用程式 更新應用程式 AWS CloudFormation	60000
MinPauseBetweenCheckpoints	可修改	建立應用程式 更新應用程式 AWS CloudFormation	5000
未對齊的檢查點	可修改	支援案例	False

設定	可以修改嗎？	方法	預設值
並行檢查點的數量	不可修改	N/A	1
檢查點模式	不可修改	N/A	恰好一次
檢查點保留政策	不可修改	N/A	失敗時
檢查點逾時	不可修改	N/A	60 分鐘
最大檢查點	不可修改	N/A	1
重新啟動策略	不可修改	N/A	固定延遲，每 10 秒無限次重試。
檢查點和儲存點位置	不可修改	N/A	我們將持久的檢查點和儲存點資料儲存到服務擁有的 S3 儲存貯體。
狀態後端記憶體閾值	不可修改	N/A	1048576

儲存點

依預設，從儲存點還原時，恢復操作會嘗試將儲存點的所有狀態映射回您要還原的程式。如果您捨棄某個運算子，依預設，從具有對應於遺失運算子之資料的儲存點還原將會失敗。您可以透過將應用程式 FlinkRunConfiguration 的 AllowNonRestoredState 參數設定為 `true`，以允許操作成功。這將允許恢復操作跳過無法對應至新程式的狀態。

如需詳細資訊，請參閱《Apche Flink 文件》 https://nightlies.apache.org/flink/flink-docs-release-1.15/中的允許非還原的狀態。

堆積大小

Managed Service for Apache Flink 會為每個 KPU 分配 3 GiB 的 JVM 堆積，並為原生程式碼配置保留 1 GiB。如需增加應用程式容量的相關資訊，請參閱the section called “擴展”。

如需 JVM 堆積的詳細資訊，請參閱《Apache Flink 文件》中的組態https://nightlies.apache.org/flink/flink-docs-release-1.15/。

緩衝區消脹

緩衝區消脹可以幫助應用程式處理高背壓。如果應用程式遇到檢查點/儲存點失敗，啟用此功能可能會很有用。要做到這一點，可請求支援案例。

如需詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的緩衝區消脹機制。

可修改的 Flink 組態屬性

以下是您可以使用支援案例修改的 Flink 組態設定。您可以一次修改多個屬性，也可以透過指定應用程式前綴來同時修改多個應用程式。如果要修改此清單以外的其他 Flink 組態屬性，請在您的案例中指定確切的屬性。

容錯能力

```
restart-strategy:

restart-strategy.fixed-delay.delay:
```

檢查點和狀態後端

```
state.backend:

state.backend.fs.memory-threshold:

state.backend.incremental:
```

檢查點

```
execution.checkpointing.unaligned:
```

RocksDB 原生指標

RockSDB 原生指標不會運送到 CloudWatch。啟用後，您可以從 Flink 儀表板或使用自訂工具從 Flink REST API 存取這些指標。

Managed Service for Apache Flink 可讓客戶使用 CreateApplicationPresignedUrl API，以唯讀模式存取最新的 Flink REST API (或您正在使用的受支援版本)。此 API 由 Flink 自己的儀表板使用，但也可以供自訂監控工具使用。

```
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:
```

進階狀態後端選項

```
state.storage.fs.memory-threshold:
```

完整 TaskManager 選項

```
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:

taskmanager.network.memory.buffer-debloat.enabled:

taskmanager.network.memory.buffer-debloat.period:

taskmanager.network.memory.buffer-debloat.samples:

taskmanager.network.memory.buffer-debloat.threshold-percentages:
```

記憶體組態

```
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:
```

用戶端

```
client.timeout:
```

進階叢集選項

```
cluster.intercept-user-system-exit:

cluster.processes.halt-on-fatal-error:
```

檔案系統組態

```
fs.s3.connection.maximum:

fs.s3a.connection.maximum:
```

```
fs.s3a.threads.max:

s3.upload.max.concurrent.uploads:
```

進階容錯選項

```
heartbeat.timeout:

jobmanager.execution.failover-strategy:
```

記憶體組態

```
jobmanager.memory.heap.size:
```

指標

```
metrics.latency.interval:
```

REST 端點和用戶端進階選項

```
rest.flamegraph.enabled:

rest.server.numThreads:
```

進階 SSL 安全性選項

```
security.ssl.internal.handshake-timeout:
```

進階排程選項

```
slot.request.timeout:
```

Flink Web UI 進階選項

```
web.timeout:
```

檢視已設定的 Flink 屬性

您可以透過 Apache Flink 儀表板檢視您自己設定或請求透過支援案例修改的 Apache Flink 屬性，並遵循下列步驟進行操作：

1. 前往 Flink 儀表板
2. 在左側導覽窗格中選擇作業管理員。
3. 選擇組態以檢視 Flink 屬性的清單。

設定 Managed Service for Apache Flink 存取 Amazon VPC 中的資源

您可以設定 Managed Service for Apache Flink 應用程式連線到您帳戶中虛擬私有雲端 (VPC) 中的私有子網路。使用 Amazon Virtual Private Cloud (Amazon VPC) 為資料庫、快取執行個體或內部服務等資源建立私有網路。將應用程式連線到 VPC 以在執行期間存取私有資源。

本主題包含下列章節：

- Amazon VPC 概念
- VPC 應用程式許可
- VPC 連線的 Managed Service for Apache Flink 的網際網路和服務存取
- Managed Service for Apache Flink VPC API
- 範例：使用 VPC 存取 Amazon MSK 叢集中的資料

Amazon VPC 概念

Amazon VPC 是 Amazon EC2 的網路層。如果您是 Amazon EC2 的新手，請參閱《適用於 Linux 執行個體的 Amazon EC2 使用者指南》中的什麼是 Amazon EC2？，以取得簡要概觀。

以下是 VPC 的重要概念：

- Virtual Private Cloud (VPC) 是您的 AWS 帳戶所專用的虛擬網路。
- 子網是您的 VPC 中的 IP 地址範圍。
- 「路由表」包含一組名為路由的規則，用來判斷網路流量的方向。
- 網際網路閘道是一種水平擴展、備援且高可用性的 VPC 元件，允許 VPC 中執行個體與網際網路之間的通訊。因此不會對網路流量強加可用性風險或頻寬限制。
- VPC 端點可讓您將 VPC 私下連線至支援的 AWS 服務以及具有 PrivateLink 功能的 VPC 端點服務，而不需要網際網路閘道、NAT 裝置、VPN 連線或 AWS Direct Connect 連線。VPC 中的執行個體不需要公有 IP 地址，即可與服務中的資源通訊。VPC 與另一個服務之間的流量都會保持在 Amazon 網路的範圍內。

如需 Amazon VPC 服務的詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html。

Managed Service for Apache Flink 會在應用程式的 VPC 組態中提供的其中一個子網路中建立彈性網路介面。VPC 子網路中建立的彈性網路介面數目可能會有所不同，取決於應用程式平行處理層級及其每個 KPU 的平行處理層級。如需應用程式擴展的詳細資訊，請參閱擴展。

> **Note**
>
> SQL 應用程式不支援 VPC 組態。

> **Note**
>
> Managed Service for Apache Flink 服務管理具有 VPC 組態之應用程式的檢查點和快照狀態。

VPC 應用程式許可

本節說明您的應用程式在使用 VPC 時需要的許可政策。如需關於使用許可政策的詳細資訊，請參閱 Amazon Managed Service for Apache Flink 的身分和存取管理。

下列許可政策會授予您的應用程式與 VPC 互動的必要許可。若要使用此許可政策，請將其新增至應用程式的執行角色。

Amazon VPC 的存取許可政策

```
{
  "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**
>
> 當您使用主控台 (例如 CloudWatch Logs 或 Amazon VPC) 指定應用程式資源時，主控台會修改您的應用程式執行角色，以授予存取這些資源的許可。只有在不使用主控台的情況下建立應用程式時，才需要手動修改應用程式的執行角色。

VPC 連線的 Managed Service for Apache Flink 的網際網路和服務存取

當您將 Managed Service for Apache Flink 應用程式連線至您帳戶的 VPC 時，除非 VPC 提供存取權，否則該應用程式無法存取網際網路。如果應用程式需要網際網路存取，必須符合下列條件：

- Managed Service for Apache Flink 應用程式只能使用私有子網路進行設定。
- VPC 必須在公有子網路中包含 NAT 閘道或執行個體。
- 從私有子網路到公有子網路 NAT 閘道之間必須存在路由以傳輸傳出流量。

> **Note**
>
> 多項服務提供 VPC 端點。您可以使用 VPC 端點從 VPC 內部連線至 Amazon 服務，而不需要存取網際網路。

子網路是公有還是私有，取決於其路由表。每個路由表都有一個預設路由，決定具有公用目的地之封包的下一個躍點。

- 對於私有子網路：預設路由指向 NAT 閘道 (nat-...) 或 NAT 執行個體 (eni-...)。
- 對於公有子網路：預設路由指向網際網路閘道 (igw-...)。

使用一個公有子網路 (使用 NAT) 和一或多個私有子網路設定 VPC 後，請執行下列動作以識別您的私有和公有子網路：

- 在 VPC 主控台的導覽窗格中，選擇子網路。
- 選取子網路，然後選擇路由表標籤。驗證預設路由：
  - 公有子網路：目的地：0.0.0.0/0，目標：igw-…
  - 私有子網路：目的地：0.0.0.0/0，目標：nat-… 或 eni-…

在 Managed Service for Apache Flink 應用程式與私有子網路之間建立關聯：

- 前往 https://console.aws.amazon.com/flink 開啟 Managed Service for Apache Flink 主控台
- 在 Managed Service for Apache Flink 頁面上，選擇您的應用程式，然後選擇應用程式詳細資料。
- 在應用程式頁面中，選擇設定。
- 在 VPC 連線區段中，選擇要與應用程式建立關聯的 VPC。選擇希望應用程式用來存取 VPC 資源的與 VPC 相關聯的子網路和安全群組。
- 選擇更新。

相關資訊

建立含公有子網路和私有子網路的 VPC

NAT 閘道基本概念

Managed Service for Apache Flink VPC API

您可以使用下列 Managed Service for Apache Flink API 作業來管理應用程式的 VPC。如需關於使用 Managed Service for Apache Flink API 的資訊，請參閱API 範例程式碼。

CreateApplication

使用 CreateApplication 動作可在建立期間將 VPC 組態新增至應用程式。

`CreateApplication` 動作的下列請求程式碼範例包括建立應用程式時的 VPC 組態：

```
{
  "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

使用 AddApplicationVpcConfiguration 動作可在建立 VPC 組態之後將其新增至應用程式。

AddApplicationVpcConfiguration 動作的下列範例請求程式碼可將 VPC 組態新增至現有的應用程式：

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

```
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
   }
}
```

DeleteApplicationVpcConfiguration

使用 DeleteApplicationVpcConfiguration 動作可從應用程式中移除 VPC 組態。

AddApplicationVpcConfiguration 動作的下列範例請求程式碼可將現有的 VPC 組態從應用程式中移除：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfigurationId": "1.1"
}
```

UpdateApplication

使用 UpdateApplication 動作可一次更新應用程式的所有 VPC 組態。

UpdateApplication 動作的下列範例請求程式碼可更新應用程式的所有 VPC 組態：

```
{
   "ApplicationConfigurationUpdate": {
      "VpcConfigurationUpdates": [
         {
            "SecurityGroupIdUpdates": [ "sg-0123456789abcdef0" ],
            "SubnetIdUpdates": [ "subnet-0123456789abcdef0" ],
            "VpcConfigurationId": "2.1"
         }
      ]
   },
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9
}
```

範例：使用 VPC 存取 Amazon MSK 叢集中的資料

如需關於如何在 VPC 中存取 Amazon MSK 叢集資料的完整教學課程，請參閱MSK 複寫。

Managed Service for Apache Flink 疑難排解

以下內容可協助您對 Amazon Managed Service for Apache Flink 中可能遇到的問題進行疑難排解。

主題

開發問題疑難排解

主題

Apache Flink 火焰圖

火焰圖在 Managed Service for Apache Flink 版本支援的應用程式上預設為啟用。如果讓火焰圖保持開啟狀態，可能會影響應用程式效能，如 Flink 文檔中所述。

如果要為您的應用程式停用火焰圖，請建立一個案例，以請求將其為您的應用程式 ARN 停用。如需詳細資訊，請參閱 AWS 支援中心。

EFO 連接器 1.15.2 的憑證提供者問題

Kinesis Data Streams EFO 連接器截止 1.15.2 版本都存在一個已知問題，其中的 `FlinkKinesisConsumer` 不遵守 `Credential Provider` 組態。由於該問題，有效組態被忽略，導致使用 AUTO 憑證提供者。使用 EFO 連接器跨帳戶存取 Kinesis 時，這可能會導致問題。

若要解決此錯誤，請使用 EFO 連接器 1.15.3 版或更新版本。

應用程式使用不支援的 Kinesis 連接器

如果應用程式使用綁定在應用程式 JAR 或存檔 (ZIP) 中的不支援 Kinesis 連接器版本 (1.15.2 之前版本)，適用於 Apache Flink 1.15 版的 Managed Service for Apache Flink 將會自動拒絕應用程式啟動或更新。

拒絕錯誤

透過以下方式提交建立/更新應用程式的呼叫時，將看到以下錯誤：

```
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
```

要修復的步驟

- 更新應用程式的 `flink-connector-kinesis` 相依項。如果使用 Maven 作為專案的建置工具，請按照 更新 Maven 相依項 操作。如果使用 Gradle，請按照 更新 Gradle 相依項 操作。
- 重新封裝應用程式。
- 上傳至 Amazon S3 儲存貯體。
- 使用剛上傳到 Amazon S3 儲存貯體的修訂後應用程式重新提交建立/更新應用程式的請求。
- 如果繼續看到相同的錯誤訊息，請重新檢查應用程式相依性。如果問題仍然存在，請建立一個支持票證。

更新 Maven 相依項

1. 開啟專案的 `pom.xml`。

2. 尋找專案的相依項。他們看起來如下所示：

```
<project>

    ...

    <dependencies>

        ...

        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-connector-kinesis</artifactId>
        </dependency>

        ...

    </dependencies>

    ...

</project>
```

3. 將 `flink-connector-kinesis` 更新至 1.15.2 或更新版本。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 相依項

1. 開啟專案的 `build.gradle` (或針對 Kotlin 應用程式的 `build.gradle.kts`)。
2. 尋找專案的相依項。他們看起來如下所示：

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis")

    ...

}

...
```

3. 將 `flink-connector-kinesis` 更新至 1.15.2 或更新版本。例如：

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis:1.15.2")

    ...

}

...
```

編譯錯誤：「無法解析專案的相依項」

若要編譯 Managed Service for Apache Flink 範例應用程式，必須先下載並編譯 Apache Flink Kinesis 連接器，並將其新增至本機 Maven 儲存庫。如果連接器尚未新增至儲存庫，則會出現如下編譯錯誤：

```
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
```

若要解決此錯誤，必須從 https://flink.apache.org/downloads.html 為連接器下載 Apache Flink 來源程式碼 1.8.2 版本。如需關於如何下載、編譯和安裝 Apache Flink 來源程式碼的指示，請參閱the section called “將 Apache Flink Kinesis 串流連接器與 Apache Flink 先前版本搭配使用”。

無效選擇：「kinesisanalyticsv2」

若要使用 Managed Service for Apache Flink API v2，需要最新版本的 AWS Command Line Interface (AWS CLI)。

如需關於升級 AWS CLI 的資訊，請參閱《AWS Command Line Interface 使用者指南》中的安裝 AWS Command Line Interface。

UpdateApplication 動作未重新載入應用程式程式碼

如果未指定 S3 物件版本，UpdateApplication動作將不會以相同的檔案名稱重新載入應用程式程式碼。若要使用相同的檔案名稱重新載入應用程式程式碼，請在 S3 儲存貯體上啟用版本控制，然後使用 `ObjectVersionUpdate` 參數指定新的物件版本。如需關於在 S3 儲存貯體中啟用物件版本控制的詳細資訊，請參閱啟用和停用物件版本控制。

S3 StreamingFileSink FileNotFoundExceptions

如果缺少由儲存點參照的進行中分段檔案，則從快照開始時，Managed Service for Apache Flink 應用程式可能會遇到進行中的分段檔案 `FileNotFoundException`。發生此失敗模式時，Managed Service for Apache Flink 應用程式的運算子狀態通常不可復原，必須在不使用快照的情況下使用 `SKIP_RESTORE_FROM_SNAPSHOT` 重新啟動。請參閱以下範例 stacktrace：

```
java.io.FileNotFoundException: No such file or directory: s3://your-s3-bucket/pathj/
INSERT/2023/4/19/7/_part-2-1234_tmp_12345678-1234-1234-1234-123456789012
        at
 org.apache.hadoop.fs.s3a.S3AFileSystem.s3GetFileStatus(S3AFileSystem.java:2231)
```

```
        at
 org.apache.hadoop.fs.s3a.S3AFileSystem.innerGetFileStatus(S3AFileSystem.java:2149)
        at
 org.apache.hadoop.fs.s3a.S3AFileSystem.getFileStatus(S3AFileSystem.java:2088)
        at org.apache.hadoop.fs.s3a.S3AFileSystem.open(S3AFileSystem.java:699)
        at org.apache.hadoop.fs.FileSystem.open(FileSystem.java:950)
        at
 org.apache.flink.fs.s3hadoop.HadoopS3AccessHelper.getObject(HadoopS3AccessHelper.java:98)
        at
 org.apache.flink.fs.s3.common.writer.S3RecoverableMultipartUploadFactory.recoverInProgressPart
...
```

Flink StreamingFileSink 會將記錄寫入 F FileSystem link 抽象支援的檔案系統。鑒於傳入串流可以無限制，資料會組織成有限大小的分段檔案，並在寫入資料時添加新檔案。分段生命週期和輪替政策可決定分段檔案的時間、大小和命名。

> **Note**
>
> 如需詳細資訊，請參閱分段檔案生命週期。

在檢查點和儲存點 (快照) 期間，所有待處理檔案都會重新命名並遞交。但是，進行中的分段檔案不會遞交而是重新命名，且其參考會保留在還原作業時要使用的檢查點或儲存點中繼資料內。這些進行中的分段檔案最終會輪替為「待處理」狀態，由後續檢查點或儲存點重新命名或遞交。

以下是遺失進行中分段檔案的根本原因和緩解措施：

- 用於啟動 Apache Flink 應用程式的受管服務的過時快照 — 只有應用程式停止或更新時所拍攝的最新系統快照才能用於啟動 Amazon S3 的 Apache Flink 應用程式的受管服務。 StreamingFileSink若要避免這類失敗，請使用最新的系統快照。
  - 例如，當您選擇使用 CreateSnapshot 建立的快照，而不是在停止或更新期間系統觸發的快照時，就會發生這種情況。較舊快照的儲存點會保留對進行中零件檔案的 out-of-date 參考，該檔案已由後續檢查點或儲存點重新命名並認可。
  - 當選取上午系統快照觸發自非最新的停止/更新事件時，也可能發生這種情況。其中一個範例是已停用系統快照但已設定 RESTORE_FROM_LATEST_SNAPSHOT 的應用程式。一般而言，適用於 Amazon S3 的 Apache Flink 應用程式的受管服務 StreamingFileSink 應始終啟用和RESTORE_FROM_LATEST_SNAPSHOT設定系統快照。
- 移除進行中的分段檔案 — 由於進行中的分段檔案位於 S3 儲存貯體中，因此可以由其他可存取該儲存貯體的元件或參與者移除。

  - 當您停止應用程式太長時間，且 S3 儲存貯 MultiPartUpload體生命週期政策移除了應用程式儲存點所參考的進行中零件檔案時，就可能會發生這種情況。若要避免此類失敗，請確保 S3 儲存貯體 MPU 生命週期政策針對您的使用案例涵蓋了足夠長的期間。
  - 當進行中的分段檔案已手動移除或由系統的其他元件移除時，也會發生這種情況。若要避免此類失敗，請確定進行中的分段檔案不會被其他參與者或元件移除。
- 在儲存點之後觸發自動檢查點的競爭情形 — 這會影響 Managed Service for Apache Flink 1.13 及以下版本。此問題已在 Managed Service for Apache Flink 1.15 版本中修正；請將您的應用程式移轉至 Managed Service for Apache Flink 1.15 版本，以避免再次發生。我們還建議從遷移 StreamingFileSink 到 FileSink.
  - 應用程式停止或更新後，Managed Service for Apache Flink 會觸發儲存點，並透過兩個步驟停止應用程式。如果在這兩個步驟之間觸發了自動檢查點，則儲存點將無法使用，因為其進行中分段檔案將會重新命名並可能遞交。

FlinkKafkaConsumer 與保存點停止問題

使用舊版時，如 FlinkKafkaConsumer 果您啟用了系統快照，您的應用程序可能會卡在更新，停止或擴展中。此問題沒有已發佈的修正程式可用，因此建議您升級KafkaSource至新版本以減輕此問題。

如果您正在啟用快照的情況下使用 `FlinkKafkaConsumer`，Flink 作業可能會使用儲存點 API 請求處理停止，`FlinkKafkaConsumer` 可能會失敗並報告執行時間錯誤 `ClosedException`。在這些情況下，Flink 應用程式會卡住，並顯示為失敗的檢查點。

FLINK 1.15 非同步接收器死鎖

Apache Flink 實作 AsyncSink 介面的連AWS接器存在一個已知的問題。這會影響使用具有下列連接器的 Flink 1.15 的應用程式：

- 對於 Java 應用程式：
  - KinesisStreamsSink – `org.apache.flink:flink-connector-kinesis`
  - KinesisStreamsSink – `org.apache.flink:flink-connector-aws-kinesis-streams`
  - KinesisFirehoseSink – `org.apache.flink:flink-connector-aws-kinesis-firehose`
  - DynamoDbSink – `org.apache.flink:flink-connector-dynamodb`

- 快速 SQL/資料表 API/Python 應用程式：
  - kinesis – `org.apache.flink:flink-sql-connector-kinesis`

- kinesis – `org.apache.flink:flink-sql-connector-aws-kinesis-streams`
- firehose – `org.apache.flink:flink-sql-connector-aws-kinesis-firehose`
- dynamodb – `org.apache.flink:flink-sql-connector-dynamodb`

受影響的應用程式會出現下列徵狀：

- FLINK 作業處於 RUNNING 狀態，但未在處理資料；
- 沒有作業重新啟動;
- 檢查點逾時。

該問題由 AWS SDK 中的錯誤引起，導致在使用非同步 HTTP 用戶端時不會向呼叫者顯示某些錯誤。這會導致接收器在檢查點排清操作期間無限期等待「進行中請求」完成。

從 AWS SDK 2.20.144 版開始，此問題已修正。

以下是如何更新受影響的連接器，以便在應用程式中使用新版 AWS SDK 的指示：

主題

- 更新 Java 應用程式
- 更新 Python 應用程式

更新 Java 應用程式

請依照下列程序更新 Java 應用程式：

flink-connector-kinesis

如果應用程式使用 `flink-connector-kinesis`：

Kinesis 連接器使用遮蔽將某些相依項 (包括 AWS SDK) 封裝到連接器 jar 中。若要更新 AWS SDK 版本，請使用下列程序來取代這些遮蔽的類別：

Maven

1. 將 Kinesis 連接器和所需的 AWS SDK 模組新增為專案相依項。
2. 設定 `maven-shade-plugin`：

    a. 在複製 Kinesis 連接器 jar 的內容時，新增篩選器以排除遮蔽的 AWS SDK 類別。

b. 新增重新放置規則，以將更新的 AWS SDK 類別移至 Kinesis 連接器所預期的套件中。

pom.xml

```
<project>
    ...
    <dependencies>
        ...
        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-connector-kinesis</artifactId>
            <version>1.15.4</version>
        </dependency>

        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>kinesis</artifactId>
            <version>2.20.144</version>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>netty-nio-client</artifactId>
            <version>2.20.144</version>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>sts</artifactId>
            <version>2.20.144</version>
        </dependency>
        ...
    </dependencies>
    ...
    <build>
        ...
        <plugins>
            ...
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>3.1.1</version>
                <executions>
                    <execution>
```

```
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                        <configuration>
                            ...
                            <filters>
                                ...
                                <filter>
                                    <artifact>org.apache.flink:flink-connector-
kinesis</artifact>
                                    <excludes>
                                        <exclude>org/apache/flink/kinesis/
shaded/software/amazon/awssdk/**</exclude>
                                        <exclude>org/apache/flink/kinesis/
shaded/org/reactivestreams/**</exclude>
                                        <exclude>org/apache/flink/kinesis/
shaded/io/netty/**</exclude>
                                        <exclude>org/apache/flink/kinesis/
shaded/com/typesafe/netty/**</exclude>
                                    </excludes>
                                </filter>
                                ...
                            </filters>
                            <relocations>
                                ...
                                <relocation>
                                    <pattern>software.amazon.awssdk</pattern>

 <shadedPattern>org.apache.flink.kinesis.shaded.software.amazon.awssdk</
shadedPattern>
                                </relocation>
                                <relocation>
                                    <pattern>org.reactivestreams</pattern>

 <shadedPattern>org.apache.flink.kinesis.shaded.org.reactivestreams</
shadedPattern>
                                </relocation>
                                <relocation>
                                    <pattern>io.netty</pattern>

 <shadedPattern>org.apache.flink.kinesis.shaded.io.netty</shadedPattern>
                                </relocation>
                                <relocation>
```

```
                                    <pattern>com.typesafe.netty</pattern>

 <shadedPattern>org.apache.flink.kinesis.shaded.com.typesafe.netty</
shadedPattern>
                                </relocation>
                                ...
                            </relocations>
                            ...
                        </configuration>
                    </execution>
                </executions>
            </plugin>
            ...
        </plugins>
        ...
    </build>
</project>
```

Gradle

1. 將 Kinesis 連接器和所需的 AWS SDK 模組新增為專案相依項。
2. 調整 shadowJar 組態：

   a. 在複製 Kinesis 連接器 jar 的內容時，排除遮蔽的 AWS SDK 類別。

   b. 將更新的 AWS SDK 類別重新放置到 Kinesis 連接器所預期的套件中。

   build.gradle

```
...
dependencies {
    ...
    flinkShadowJar("org.apache.flink:flink-connector-kinesis:1.15.4")

    flinkShadowJar("software.amazon.awssdk:kinesis:2.20.144")
    flinkShadowJar("software.amazon.awssdk:sts:2.20.144")
    flinkShadowJar("software.amazon.awssdk:netty-nio-client:2.20.144")
    ...
}
...
shadowJar {
    configurations = [project.configurations.flinkShadowJar]
```

```
    exclude("org/apache/flink/kinesis/shaded/software/amazon/awssdk/**/*.class")
    exclude("org/apache/flink/kinesis/shaded/org/reactivestreams/**/*.class")
    exclude("org/apache/flink/kinesis/shaded/io/netty/**/*.class")
    exclude("org/apache/flink/kinesis/shaded/com/typesafe/netty/**/*.class")

    relocate("software.amazon.awssdk",
 "org.apache.flink.kinesis.shaded.software.amazon.awssdk")
    relocate("org.reactivestreams",
 "org.apache.flink.kinesis.shaded.org.reactivestreams")
    relocate("io.netty", "org.apache.flink.kinesis.shaded.io.netty")
    relocate("com.typesafe.netty",
 "org.apache.flink.kinesis.shaded.com.typesafe.netty")
}
...
```

其他受影響連接器

若應用程式使用其他受影響的連接器：

為了更新 AWS SDK 版本，應在專案建置組態中強制執行 SDK 版本。

Maven

將 AWS SDK 物料清單 (BOM) 新增至 pom.xml 檔案的相依性管理區段，以為專案強制執行 SDK 版本。

pom.xml

```
<project>
    ...
    <dependencyManagement>
        <dependencies>
            ...
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>2.20.144</version>
                <scope>import</scope>
                <type>pom</type>
            </dependency>
            ...
        </dependencies>
```

```
    </dependencyManagement>
    ...
</project>
```

Gradle

在 AWS SDK 物料清單 (BOM) 上新增平台相依性，以為專案強制執行 SDK 版本。這需要 Gradle 5.0 或更新版本：

build.gradle

```
...
dependencies {
    ...
    flinkShadowJar(platform("software.amazon.awssdk:bom:2.20.144"))
    ...
}
...
```

更新 Python 應用程式

Python 應用程式可以透過 2 種不同的方式使用連接器：將連接器和其他 Java 相依項作封裝為單個 uber-jar 的一部分，或直接使用連接器 jar。若要修正受非同步接收器死鎖影響的應用程式：

- 如果應用程式使用 uber jar，請依照 更新 Java 應用程式 的指示操作。
- 若要從來源重建連接器 jar，請使用下列步驟：

從來源建置連接器：

先決條件，類似於 Flink 建置需求：

- Java 11
- Maven 3.2.5

flink-sql-connector-kinesis

1. 下載 Flink 1.15.4 的來源程式碼：

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. 解壓縮來源程式碼：

```
tar -xvf flink-1.15.4-src.tgz
```

3. 導覽至 Kinesis 連接器目錄

```
cd flink-1.15.4/flink-connectors/flink-connector-kinesis/
```

4. 編譯並安裝連接器 jar，指定要求的 AWS SDK 版本。為了加快建置，使用 `-DskipTests` 跳過測試執行，並使用 `-Dfast` 跳過其他來源程式碼檢查：

```
mvn clean install -DskipTests -Dfast -Daws.sdkv2.version=2.20.144
```

5. 導覽至 Kinesis 連接器目錄

```
cd ../flink-sql-connector-kinesis
```

6. 編譯並安裝 sql 連接器 jar：

```
mvn clean install -DskipTests -Dfast
```

7. 產生的 jar 將在以下位置提供：

```
target/flink-sql-connector-kinesis-1.15.4.jar
```

flink-sql-connector-aws-運動流

1. 下載 Flink 1.15.4 的來源程式碼：

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. 解壓縮來源程式碼：

```
tar -xvf flink-1.15.4-src.tgz
```

3. 導覽至 Kinesis 連接器目錄

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-streams/
```

4. 編譯並安裝連接器 jar，指定要求的 AWS SDK 版本。為了加快建置，使用 -DskipTests 跳過測試執行，並使用 -Dfast 跳過其他來源程式碼檢查：

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. 導覽至 Kinesis 連接器目錄

```
cd ../flink-sql-connector-aws-kinesis-streams
```

6. 編譯並安裝 sql 連接器 jar：

```
mvn clean install -DskipTests -Dfast
```

7. 產生的 jar 將在以下位置提供：

```
target/flink-sql-connector-aws-kinesis-streams-1.15.4.jar
```

flink-sql-connector-aws-運動-火喉

1. 下載 Flink 1.15.4 的來源程式碼：

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. 解壓縮來源程式碼：

```
tar -xvf flink-1.15.4-src.tgz
```

3. 導覽至連接器目錄

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-firehose/
```

4. 編譯並安裝連接器 jar，指定要求的 AWS SDK 版本。為了加快建置，使用 -DskipTests 跳過測試執行，並使用 -Dfast 跳過其他來源程式碼檢查：

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. 導覽至 sql 連接器目錄

```
cd ../flink-sql-connector-aws-kinesis-firehose
```

6. 編譯並安裝 sql 連接器 jar：

```
mvn clean install -DskipTests -Dfast
```

7. 產生的 jar 將在以下位置提供：

```
target/flink-sql-connector-aws-kinesis-firehose-1.15.4.jar
```

flink-sql-connector-dynamodb

1. 下載 Flink 1.15.4 的來源程式碼：

```
wget https://archive.apache.org/dist/flink/flink-connector-aws-3.0.0/flink-
connector-aws-3.0.0-src.tgz
```

2. 解壓縮來源程式碼：

```
tar -xvf flink-connector-aws-3.0.0-src.tgz
```

3. 導覽至連接器目錄

```
cd flink-connector-aws-3.0.0
```

4. 編譯並安裝連接器 jar，指定要求的 AWS SDK 版本。為了加快建置，使用 `-DskipTests` 跳過測試執行，並使用 `-Dfast` 跳過其他來源程式碼檢查：

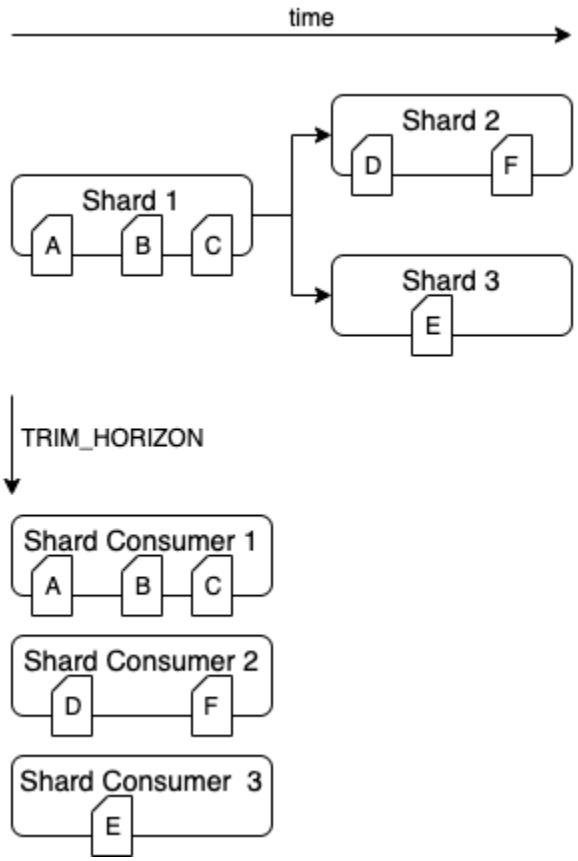
```
mvn clean install -DskipTests -Dfast -Dflink.version=1.15.4 -
Daws.sdk.version=2.20.144
```

5. 產生的 jar 將在以下位置提供：

```
flink-sql-connector-dynamodb/target/flink-sql-connector-dynamodb-3.0.0.jar
```

在重新分片期間，Amazon Kinesis Data Streams 來源處理不按順序

目前的 FlinkKinesisConsumer 實作不會在 Kinesis 碎片之間提供強有力的排序保證。這可能會導致 Kinesis Stream 重新分片期間進行 out-of-order 處理，特別是對於遇到處理延遲的 Flink 應用程式而言。在某些情況下，例如根據事件時間的 Windows 運算子，事件可能會因為產生的延遲而被捨棄。



這是開放原始碼 Flink 中的已知問題。在提供連接器修正之前，請確保您的 Flink 應用程式在重新分割期間不會落後於 Kinesis 資料串流。通過確保 Flink 應用程序可以容忍處理延遲，您可以最大程度地減少 out-of-order 處理的影響和數據丟失的風險。

執行時間疑難排解

本節包含診斷和修正 Managed Service for Apache Flink 應用程式執行時間問題的相關資訊。

主題

- 疑難排解工具
- 應用程式問題
- 應用程式重新啟動
- 輸送量太慢
- 無限制狀態增長
- I/O 綁定運算子
- Kinesis 資料串流的上游或來源限流
- 檢查點

- 檢查點逾時
- Apache Beam 應用程式檢查點失敗
- 背壓
- 資料扭曲
- 狀態扭曲
- 與不同地區的資源整合

疑難排解工具

偵測應用程式問題的主要工具是 CloudWatch 警示。您可以使用 CloudWatch 警示設定 CloudWatch 指標的閾值，以指出應用程式中的錯誤或瓶頸狀況。如需建議的 CloudWatch 警示的相關資訊，請參閱將 CloudWatch 警示與 Amazon 管理服務搭配 Apache Flink 使用。

應用程式問題

本節包含針對 Managed Service for Apache Flink 應用程式可能會遇到的錯誤情況的解決方案。

主題

- 應用程式卡在暫時狀態
- 快照建立失敗
- 無法存取 VPC 中的資源
- 寫入 Amazon S3 儲存貯體時資料遺失
- 應用程式處於 RUNNING 狀態，但未在處理資料
- 快照、應用程式更新或應用程式停止錯誤：InvalidApplicationConfigurationException
- java.nio.file.NoSuchFileException: /usr/local/openjdk-8/lib/security/cacerts

應用程式卡在暫時狀態

如果應用程式已保持暫時狀態 (STARTING、UPDATING、STOPPING 或 AUTOSCALING) 一段時間，您可以使用 StopApplication 動作停止應用程式，並將 `Force` 參數設定為 `true`。您無法強制停止 DELETING 狀態的應用程式。或者，如果應用程式處於 UPDATING 或 AUTOSCALING 狀態，您可以將其復原至先前執行的版本。復原應用程式時，它會從上次成功的快照載入狀態資料。如果應用程式沒有快照，Managed Service for Apache Flink 會拒絕復原請求。如需復原應用程式的詳細資訊，請參閱 RollbackApplication 動作。

> **Note**
>
> 強制停止應用程式可能會導致資料丟失或重複。為了防止應用程式重新啟動期間資料遺失或重複處理資料，我們建議您經常拍攝應用程式的快照。

應用程式卡住的原因如下：

- 應用程式狀態太大：應用程式狀態太大或過於持續，可能會導致應用程式在檢查點或快照操作期間卡住。檢查應用程式 `lastCheckpointDuration` 和 `lastCheckpointSize` 指標的值是否在穩定增加或異常高。
- 應用程式程式碼太大：確認您的應用程式 JAR 檔案小於 512 MB。不支援大於 512 MB 的 JAR 檔案。
- 應用程式快照建立失敗：Managed Service for Apache Flink 會在 `UpdateApplication` 或 `StopApplication` 請求期間擷取應用程式的快照。然後，服務會使用此快照狀態，並使用更新的應用程式組態還原應用程式，以提供恰好一次的處理語義。如果自動建立快照失敗，請參閱下文的快照建立失敗。
- 從快照還原失敗：如果您移除或變更應用程式更新中的運算子，並嘗試從快照還原，則如果快照包含遺失運算子的狀態資料，還原預設將會失敗。此外，應用程式將卡在 `STOPPED` 或 `UPDATING` 狀態。若要變更此行為並讓還原成功，請將應用程式 FlinkRunConfiguration 的 AllowNonRestoredState 參數變更為 `true`。這將允許恢復操作跳過無法對應至新程式的狀態資料。
- 應用程式初始化所花費的時間較長：Managed Service for Apache Flink 在等待 Flink 作業啟動時，會使用 5 分鐘的內部逾時 (軟體設定)。如果作業無法在此逾時內啟動，您將會看到如下所示的 CloudWatch 日誌：

  ```
  Flink job did not start within a total timeout of 5 minutes for application: %s under
   account: %s
  ```

  如果遇到上述錯誤，表示在 Flink 作業 `main` 方法下定義的操作需要 5 分鐘以上的時間，從而導致建立 Flink 作業的操作在 Managed Service for Apache Flink 結束時逾時。我們建議您檢查 Flink JobManager 日誌以及應用程式程式碼，查看 `main` 方法中是否預期存在此延遲。如果沒有，則需要採取措施來解決該問題，以便在 5 分鐘內完成。

您可以使用 `ListApplications` 或 `DescribeApplication` 動作來檢查應用程式狀態。

快照建立失敗

在下列情況下，Managed Service for Apache Flink 服務無法拍攝快照：

- 應用程式超過快照限制。快照的限制為 1,000。如需詳細資訊，請參閱 快照。
- 應用程式沒有存取其來源或接收器的許可。
- 應用程式的程式碼無法正常運作。
- 應用程式遇到其他組態問題。

在應用程式更新期間或停止應用程式時，如果擷取快照時發生例外狀況，請將應用程式 ApplicationSnapshotConfiguration 的 SnapshotsEnabled 屬性設定為 false，然後重試該請求。

如果應用程式的運算子未正確佈建，快照可能會失敗。如需關於調整運算子效能的資訊，請參閱運算子擴展。

應用程式回到正常狀態之後，建議您將應用程式的 SnapshotsEnabled 屬性設定為 true。

無法存取 VPC 中的資源

如果應用程式使用在 Amazon VPC 上執行的 VPC，請執行以下操作以確認應用程式是否可以存取其資源：

- 檢查 CloudWatch 日誌中是否有下列錯誤。此錯誤表示應用程式無法存取 VPC 中的資源：

```
org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after
 60000 ms.
```

  如果您看到此錯誤，請確認您的路由表設定正確，並且連接器具有正確的連線設定。

  如需設定和分析 CloudWatch 日誌的相關資訊，請參閱記錄和監控。

寫入 Amazon S3 儲存貯體時資料遺失

使用 Apache Flink 1.6.2 版將輸出寫入 Amazon S3 儲存貯體時，可能會發生一些資料遺失。我們建議您在直接使用 Amazon S3 進行輸出時，使用 Apache Flink 最新支援版本。若要使用 Apache Flink 1.6.2 寫入 Amazon S3 儲存貯體，我們建議使用 Kinesis Data Firehose。如需 Kinesis Data Firehose 搭配 Managed Service for Apache Flink 使用的詳細資訊，請參閱Kinesis Data Firehose 接收器。

應用程式處於 RUNNING 狀態，但未在處理資料

您可以使用 `ListApplications` 或 `DescribeApplication` 動作來檢查應用程式狀態。如果應用程式進入 RUNNING 狀態，但未在將資料寫入接收器，您可以透過將 Amazon CloudWatch 日誌串流新增至應用程式來解決該問題。如需詳細資訊，請參閱 使用應用程式 CloudWatch 記錄選項。日誌串流包含可用於對應用程式問題進行疑難排解的訊息。

快照、應用程式更新或應用程式停止錯誤：InvalidApplicationConfigurationException

在快照操作期間或在建立快照的操作 (例如更新或停止應用程式) 期間，可能會發生如下錯誤：

```
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 Managed Service for
 Apache Flink console or by updating
the ApplicationSnapshotConfiguration through the AWS SDK
```

應用程式無法建立快照時，會發生此錯誤。

如果在快照操作或建立快照的操作期間遇到此錯誤，請執行下列動作：

- 為應用程式停用快照。您可以在 Managed Service for Apache Flink 主控台中執行此操作，也可以使用 UpdateApplication 動作的 `SnapshotsEnabledUpdate` 參數。
- 調查無法建立快照的原因。如需詳細資訊，請參閱 應用程式卡在暫時狀態。
- 當應用程式恢復正常狀態時，再重新啟用快照。

java.nio.file.NoSuchFileException: /usr/local/openjdk-8/lib/security/cacerts

SSL 信任存放區的位置已在先前的部署中更新。為 `ssl.truststore.location` 參數改用下列值：

```
/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
```

應用程式重新啟動

如果您的應用程式運作狀況不正常，其 Apache Flink 作業就會持續失敗並重新啟動。本節說明此狀況的徵狀和疑難排解步驟。

徵狀

這種情況可能有下列徵狀：

- `FullRestarts` 指標不為零。此指標代表自您啟動應用程式後，應用程式作業已重新啟動的次數。
- `Downtime` 指標不為零。此指標代表應用程式處於 `FAILING` 或 `RESTARTING` 狀態的毫秒數。
- 應用程式日誌包含狀態變更 (變更為 `RESTARTING` 或 `FAILED`)。您可以使用下列 CloudWatch Logs Insights 查詢，來查詢您的應用程式日誌中是否有這些狀態變更：分析錯誤：應用程式任務相關的失敗

原因與解決方案

下列情況可能會導致您的應用程式變得不穩定並重複重新啟動：

- 運算子擲出例外狀況：如果应用程式運算子中的任何例外狀況未得到處理，則应用程式將容錯移轉 (透過解譯運算子無法處理失敗)。應用程式會從最新的檢查點重新啟動，以維護「只有一次」處理語義。因此，`Downtime` 在這些重新啟動期間不為零。為了防止這種情況發生，我們建議您處理應用程式程式碼中的任何可重試的例外狀況。

  您可以查詢應用程式日誌中是否包含從 `RUNNING` 到`FAILED` 的應用程式狀態變更，以調查此情況的原因。如需詳細資訊，請參閱the section called “分析錯誤：應用程式任務相關的失敗”。
- Kinesis Data Streams 未正確佈建：如果應用程式的來源或接收器是 Kinesis 資料串流，請檢查串流的指標看是否有 `ReadProvisionedThroughputExceeded` 或 `WriteProvisionedThroughputExceeded` 錯誤。

  如果看到這些錯誤，您可以增加串流的碎片數目，以增加 Kinesis 串流的可用輸送量。如需詳細資訊，請參閱如何變更 Kinesis Data Streams 中的開放碎片數目？
- 其他來源或接收器未正確佈建或不可用：確認應用程式是否正確佈建來源和接收器。檢查應用程式中使用的任何來源或接收器 (例如其他 AWS 服務或外部來源或目的地) 是否已妥善佈建、未遇到讀取或寫入限流，或是定期無法使用。

  如果您遇到相依服務的輸送量相關問題，請增加這些服務的可用資源，或調查任何錯誤或無法使用的原因。

- 運算子未正確佈建：如果應用程式中其中一個運算子的執行緒上的工作負載未正確分配，則該運算子可能會多載，應用程式可能會崩潰。如需關於調整運算子平行處理層級的資訊，請參閱適當管理運算子擴展。
- 應用程式失敗，出現 DaemonException：如果您使用的是 1.11 之前的 Apache Flink 版本，此錯誤會出現在應用程式日誌中。您可能需要升級至更新版本的 Apache Flink，以便使用 0.14 或更新版本的 KPL。
- 應用程式失敗，出現 TimeoutException、FlinkException 或 RemoteTransportException：如果任務管理員崩潰，這些錯誤可能會出現在應用程式日誌中。如果應用程式多載，任務管理員可能會遇到 CPU 或記憶體資源壓力，導致它們失敗。

  這些錯誤可能如下所示：

  - `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`

  若要疑難排解此狀況，請檢查下列各項：

  - 檢查 CloudWatch 指標，看 CPU 或記憶體使用是否出現異常尖峰。
  - 檢查應用程式是否有輸送量問題。如需詳細資訊，請參閱效能疑難排解。
  - 檢查應用程式日誌，看是否有應用程式程式碼所引發的未處理例外狀況。
- 应用程式失敗，並顯示「JaxBanNotion Module 未找到」錯誤：如果应用程式使用 Apache Beam，但沒有正確的相依項或相依項版本，則會發生此錯誤。使用 Apache Beam 的 Managed Service for Apache Flink 應用程式必須使用以下版本的相依項：

```
<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>
```

  如果您未提供正確的 `jackson-module-jaxb-annotations` 版本作為明確的相依項，應用程式會從環境相依性載入該版本，而由於版本不相符，應用程式會在執行時間崩潰。

如需將 Apache Beam 與 Managed Service for Apache Flink 搭配使用的詳細資訊，請參閱搭配使用 CloudFormation 與 Managed Service for Apache Flink。

- 应用程式失敗，並顯示 java.io.IOException：網路緩衝區數目不足

  當應用程式為網路緩衝區配置的記憶體不足時，就會發生這種情況。網路緩衝區可協助子任務之間的通信。它們用於在透過網路傳輸之前存儲記錄，並在將其解析為記錄並移交給子任務之前存儲傳入的資料。所需的網路緩衝區數目可直接根據作業圖表的平行處理層級和複雜度擴展。有許多方法可以緩解此問題：

  - 您可以設定較低的 `parallelismPerKpu`，以便為每個子任務和網路緩衝區配置更多記憶體。請注意，降低 `parallelismPerKpu` 會增加 KPU，因此會增加成本。為了避免這種情況，您可以按相同係數降低平行處理層級來保持相同數量的 KPU。
  - 您可以減少運算子的數目或將它們鏈結起來，以減少所需的緩衝區來簡化作業圖表。
  - 否則，您可以聯絡 https://aws.amazon.com/premiumsupport/ 進行自訂網路緩衝區組態。

輸送量太慢

如果應用程式處理傳入的串流資料速度不夠快，它會效能不佳且變得不穩定。本節說明此狀況的徵狀和疑難排解步驟。

徵狀

這種情況可能有下列徵狀：

- 如果應用程式的資料來源是 Kinesis 串流，則串流的 `millisbehindLatest` 指標會持續增加。
- 如果應用程式的資料來源是 Amazon MSK 叢集，則叢集的取用者延遲指標會持續增加。如需詳細資訊，請參閱《Amazon MSK 開發人員指南》https://docs.aws.amazon.com/msk/latest/developerguide/what-is-msk.html中的取用者延遲監控。
- 如果應用程式的資料來源是其他服務或來源，請檢查任何可用的取用者延遲指標或可用資料。

原因與解決方案

造成應用程式輸送量緩慢的原因可能有很多。如果應用程式未與輸入保持一致，請檢查以下內容：

- 如果輸送量延遲急劇增加，然後逐漸減少，請檢查應用程式是否正在重新啟動。應用程式在重新啟動時會停止處理輸入，進而造成延遲急劇增加。如需關於應用程式故障的詳細資訊，請參閱應用程式重新啟動。

- 如果輸送量延遲一致，請檢查應用程式是否已進行效能最佳化。如需最佳化應用程式效能的資訊，請參閱效能疑難排解。
- 如果輸送量延遲未急劇增加，而是持續增加，並且應用程式已進行效能最佳化，則必須增加應用程式資源。如需增加應用程式資源的資訊，請參閱擴展。
- 如果應用程式從不同區域的 Kafka 叢集讀取，並且儘管取用者延遲很高，FlinkKafkaConsumer 或 KafkaSource 大多是閒置狀態 (高 idleTimeMsPerSecond 或低 CPUUtilization)，則可以增加 receive.buffer.byte 的值，例如 2097152。如需詳細資訊，請參閱自訂 MSK 組態中的「高延遲環境」一節。

如需應用程式來源中輸送量緩慢或取用者延遲增加的疑難排解步驟，請參閱效能疑難排解。

無限制狀態增長

如果應用程式未正確處置過期的狀態資訊，這些資訊會持續累積並導致應用程式效能或穩定性問題。本節說明此狀況的徵狀和疑難排解步驟。

徵狀

這種情況可以具有下列徵狀：

- lastCheckpointDuration 指標正在逐漸增加或急劇增加。
- lastCheckpointSize 指標正在逐漸增加或急劇增加。

原因與解決方案

下列情況可能會導致應用程式累積狀態資料：

- 應用程式保留狀態資料的時間超過需要的時間。
- 應用程式使用持續時間過長的視窗查詢。
- 您尚未為狀態資料設定 TTL。如需詳細資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的狀態存活期 (TTL)。
- 您正在執行的應用程式相依於 Apache Beam 2.25.0 版或更高版本。您可以透過用關鍵實驗和 use_deprecated_read 值擴充 BeamapplicationProperties，選擇退出讀取轉換的新版本。如需詳細資訊，請參閱《Apache Beam 文件》https://beam.apache.org/blog/beam-2.25.0/#highlights。

應用程式有時會面臨持續擴增的狀態大小增長，從長遠來看，這是不可持續的 (畢竟 Flink 應用程式會無限期地執行)。有時，這可以追溯至存儲狀態資料且未正確地老化舊資訊的應用程式。但是有時候，

使用者對 Flink 可以提供的東西抱有根本不合理的期望。應用程式可以在跨越數天甚至數週的大型時間範圍內使用彙總。除非使用允許增量彙總的 AggregateFunctions，否則 Flink 需要將整個視窗的事件保持在狀態。

此外，當使用進程函數來實作自訂運算子時，應用程式需要從業務邏輯不再需要的狀態中移除資料。在這種情況下，狀態存活期可用於根據處理時間自動老化資料。Managed Service for Apache Flink 正在使用增量檢查點，因此狀態 TTL 基於 RocksDB 壓縮。在壓縮操作發生之後，您只能觀察到狀態大小的實際縮減 (由檢查點大小表示)。特別是對於低於 200 MB 的檢查點大小，由於狀態到期，您不太可能觀察到任何檢查點大小縮小。儲存點則基於不包含舊資料之狀態的全新副本，因此您可以在 Managed Service for Apache Flink 中觸發快照，以強制移除過期狀態。

出於偵錯目的，停用增量檢查點以更快速地驗證檢查點大小是否確實減小或穩定 (並避免 RocksBS 壓縮的影響) 是可行的。但是，這需要提交票證給服務團隊。

I/O 綁定運算子

最好避免對資料路徑上外部系統的相依性。將參考資料集保持在狀態中，而不是查詢外部系統以富集個別事件，通常效能要高很多。不過，有時候有些相依性無法輕易移至狀態，例如，如果您想要使用 Amazon Sageemaker 上託管的機器學習模型來富集事件。

透過網絡與外部系統進行互動的運算子可能會成為瓶頸並導致背壓。強烈建議使用 AsynCIO 來實作該功能，以減少單個呼叫的等待時間並避免整個應用程序變慢。

此外，對於具有 I/O 綁定運算子的應用程序，也可以增加 Apache Flink 應用程式的 ParallelismPerKPU 設定。此設定描述應用程式每 Kinesis 處理單元 (KPU) 可執行的平行子任務數目。藉由將值從預設值 1 增加到 4，應用程式利用相同的資源 (以相同的成本)，但可擴展至 4 倍的平行處理層級。這非常適用於 I/O 綁定的應用程式，但會給不是 I/O 綁定的應用程式造成額外開銷。

Kinesis 資料串流的上游或來源限流

徵狀：應用程式遇到來自其上游來源 Kinesis 資料串流的 `LimitExceededExceptions`。

潛在原因：Apache Flink 程式庫 Kinesis 連接器的預設設定設定為從 Kinesis 資料串流來源讀取，且每次 `GetRecords` 呼叫擷取的最大記錄數目具有非常積極的預設設定。依預設，Apache Flink 設定為每次 GetRecords 呼叫擷取 10,000 條記錄 (該呼叫預設每 200 毫秒進行一次)，但針對每個碎片限制為只有 1,000 條記錄。

嘗試從 Kinesis 資料串流取用時，此預設行為可能會導致限流，從而影響應用程式的效能和穩定性。

透過檢查 CloudWatch `ReadProvisionedThroughputExceeded` 指標並且看到此指標長時間或在一段持續的時間範圍內皆大於零，可以確認這一點。

客戶透過在 Managed Service for Apache Flink 應用程式的 CloudWatch 日誌中看到持續的 `LimitExceededException` 錯誤，也可以看到這一點。

解決方案：客戶可以執行下列兩項作業之一來解決此情況：

- 降低每次 GetRecords Records 呼叫擷取的記錄數目的預設限制
- 客戶可以在其 Managed Service for Apache Flink 中啟用「自適應讀取」功能。如需關於「自適應讀取」功能的詳細資訊，請參閱 SHARD_USE_ADAPTIVE_READS

檢查點

檢查點是 Flink 用於確保應用程式狀態具有容錯能力的機制。該機制允許 Flink 在作業失敗時恢復運算子的狀態，並為應用程式提供與無故障執行相同的語義。使用 Managed Service for Apache Flink，應用程式的狀態會儲存在 RocksDB 中，這是一個內嵌式索引鍵/值存放區，可將其工作狀態保留在磁碟上。取得檢查點時，狀態也會上傳至 Amazon S3，這樣即使磁碟遺失，也可以使用檢查點來還原應用程式狀態。

如需詳細資訊，請參閱狀態快照如何運作？。

檢查點階段

對於 Flink 中的檢查點運算子子任務，有 5 個主要階段：

- 等待 [開始延遲]：Flink 使用插入串流的檢查點障礙，因此在此階段的時間是運算子等待檢查點障礙到達它的時間。
- 對齊 [對齊持續時間]：在此階段，子任務已到達一個障礙，但它正在等待來自其他輸入串流的障礙。
- 同步檢查點 [同步持續時間]：在此階段，子任務會實際拍攝運算子狀態快照，並阻止該子任務上的所有其他活動。
- 非同步檢查點 [非同步持續時間]：此階段的主要操作是子任務將狀態上傳到 Amazon S3。在此階段，子任務不再被阻止，可以處理記錄。
- 確認：這通常是一個短暫的階段，只是子任務發送確認給 JobManager 並執行任何遞交訊息 (例如，使用 Kafka 接收器)。

上述每個階段 (除了「確認」) 都對應到 Flink WebUI 中可用檢查點的持續時間指標，這可以幫助隔離長檢查點的原因。

要查看檢查點上每個可用指標的確切定義，請轉到歷史記錄標籤。

調查

調查長檢查點的持續時間時，最重要的是要確定檢查點的瓶頸，也就是說，什麼運算子和子任務正在採用最長檢查點，該子任務的哪個階段正在花費較長的時間。這可以使用作業檢查點任務下的 Flink WebUI 來確定。Flink 的 Web 介面提供了可協助調查檢查點問題的資料和資訊。如需完整明細，請參閱監控檢查點。

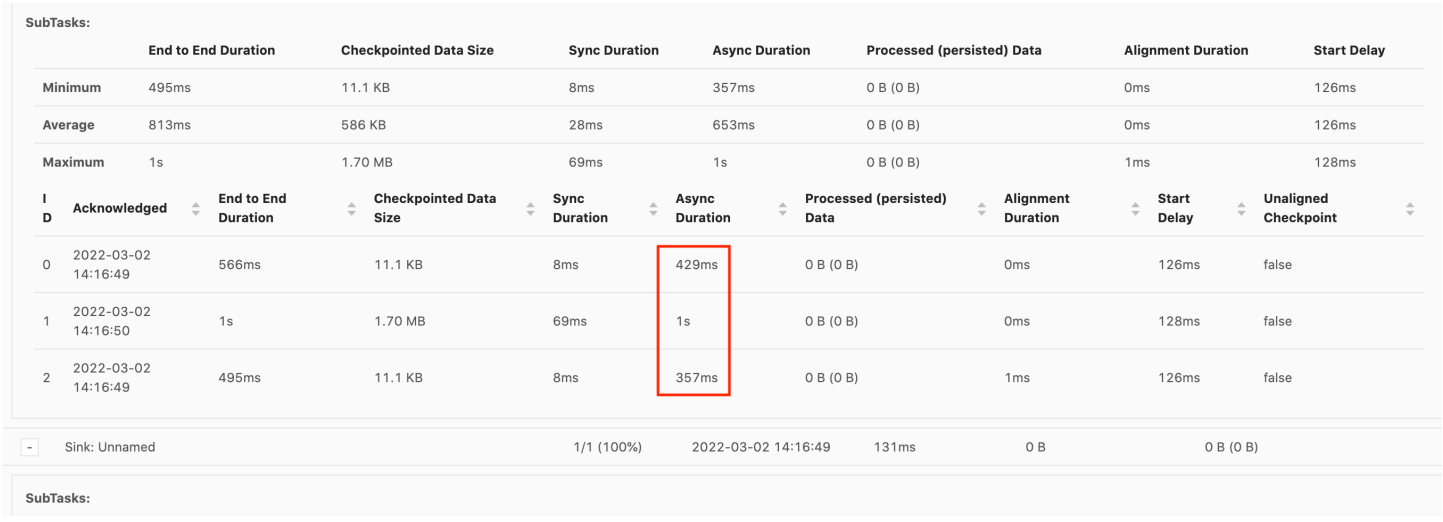
首先要注意的是作業圖表中每個運算子的端對端持續時間，以確定哪個運算子需要較長時間才能到達檢查點，需要進一步調查。根據 Flink 文件，持續時間的定義如下：

從觸發時間戳記到最近確認為止的持續時間 (如果尚未收到確認，則為 n/a)。完整檢查點的端對端持續時間由確認檢查點的最後一個子任務決定。此時間通常大於單個子任務對狀態實際執行檢查點需要的時間。

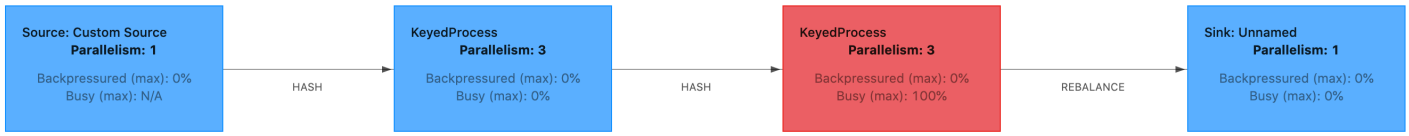
檢查點的其他持續時間還提供了有關花費時間的更精細資訊。

如果同步持續時間很高，則表示快照過程中發生了問題。在這個階段，為實作 SnapShotState 介面的類別呼叫 `snapshotState()`；這可以是使用者程式碼，所以執行緒傾印對於調查這一點會有幫助。

非同步持續時間長表明將狀態上傳到 Amazon S3 花費了大量時間。如果狀態很大，或者有許多狀態檔案正在上傳，就會發生這種情況。如果是這種情況，則值得調查應用程式如何使用狀態，並確保在可能的情況下使用 Flink 本機資料結構 (使用具有索引鍵的狀態)。Managed Service for Apache Flink 會以最小化 Amazon S3 呼叫數目的方式來設定 Flink，以確保不會變得太長。下面是某個運算子的檢查點統計資料範例。它表明，與之前的運算子檢查點統計資料相比，此運算子的非同步持續時間相對較長。



開始延遲高將表明等待檢查點障礙到達運算子花費了大部分時間。這表明應用程式正在花時間處理記錄，意味著障礙正在緩慢流經作業圖表。如果作業受到背壓或運算子經常處於忙碌狀態，通常就會發生這種情況。以下是作業圖表範例，其中第二個 KeyedProcess 運算子處於忙碌狀態。



您可以使用 Flink 火焰圖或 TaskManager 執行緒傾印來調查是什麼需要這麼長時間。一旦確定了瓶頸，就可以使用火焰圖或執行緒傾印進一步調查。

執行緒傾印

執行緒傾印是比火焰圖層級略低的另一種偵錯工具。執行緒傾印會在某個時間點輸出所有執行緒的執行狀態。Flink 接受 JVM 執行緒傾印，這是 Flink 處理序中所有執行緒的執行狀態。執行緒狀態由執行緒的堆疊追蹤以及一些附加資訊來表示。火焰圖實際上是使用快速連續採取的多個堆疊追蹤所建置。該圖形是由這些追蹤構成的可視化呈現，可讓您輕鬆地識別常見程式碼路徑。

```
"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(KeyedProcessOperat
    at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStr
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTas
    at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProces
    ...
```

以上是從 Flink UI 為單個執行緒取得的執行緒傾印的片段。第一行包含有關此執行緒的一些一般資訊，包括：

- 執行緒名稱 KeyedProcess (1/3)#0
- 執行緒優先順序 prio=5
- 唯一的執行緒 ID Id=1423
- 執行緒狀態 RUNNABLE

執行緒名稱通常會提供執行緒一般用途的資訊。運算子執行緒可以通過其名稱來識別，因為運算子執行緒與運算子具有相同的名稱，並且會指出其相關子任務，例如，KeyedProcess (1/3)#0 執行緒來自 keyedProcess 運算子，並且來自第 1 個子任務 (共 3 個)。

執行緒可以是下列幾種狀態之一：

- NEW：執行緒已建立，但尚未得到處理
- RUNNABLE：執行緒正在 CPU 上執行
- BLOCKED：執行緒正在等待另一個執行緒釋放其鎖定
- WAITING：執行緒正在使用 `wait()`、`join()` 或 `park()` 方法等待
- TIMED_WAITING：執行緒正在使用睡眠、等待、聯結或駐留方法等待，但等待時間最長。

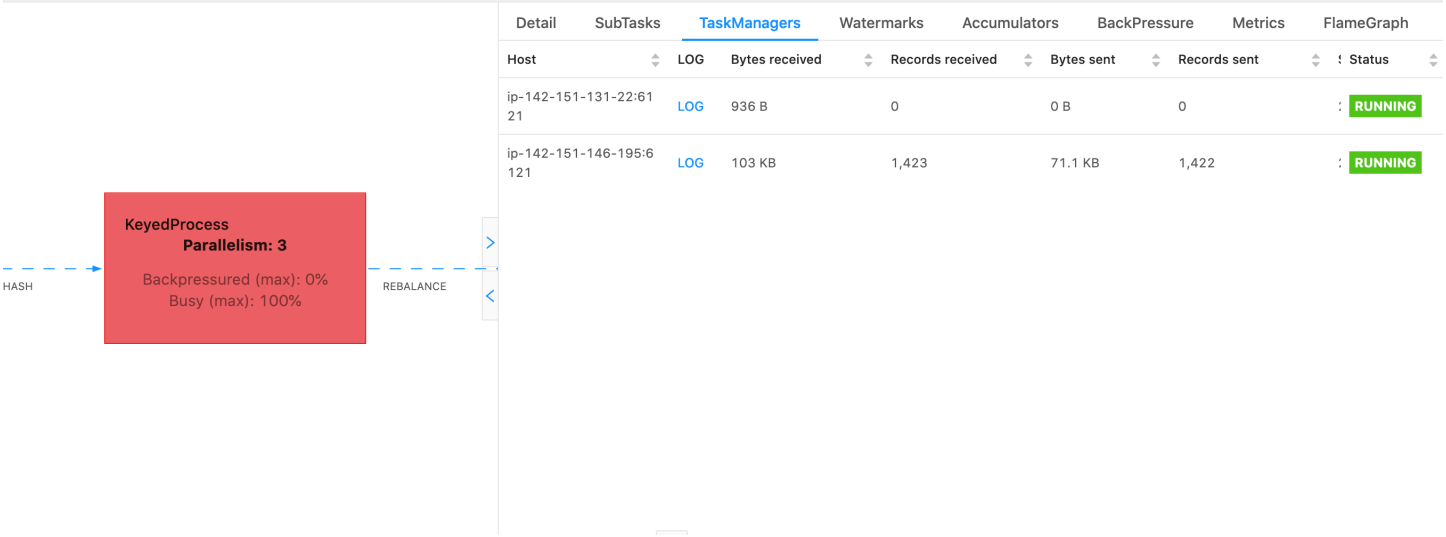
> **Note**
>
> 在 Flink 1.13 中，執行緒傾印中單一堆疊追蹤的最大深度限制為 8。

> **Note**
>
> 執行緒傾印必須是 Flink 應用程式中偵錯效能問題的最後手段，因為它們可能難以讀取，需要擷取和手動分析多個樣本。如果有可能，最好使用火焰圖。

Flink 中的執行緒傾印

在 Flink 中，透過選擇 Flink UI 左側導覽列上的任務管理員選項，選取特定任務管理員，然後瀏覽至執行緒傾印標籤，即可取得執行緒傾印。您可以下載執行緒傾印、複製到喜愛的文字編輯器 (或執行緒傾印分析器)，或直接在 Flink Web UI 的文字檢視中進行分析 (不過最後一個選項可能有點繁瑣)。

為了確定在選擇特定運算子後，要用來取得 TaskManager 選項卡執行緒傾印的任務管理器。這表明運算子正在運算子的不同子任務上執行，並且可以在不同的任務管理器上執行。



傾印將由多個堆疊追蹤組成。但是，在調查傾印時，與運算子關聯的傾印最重要。這些很容易找到，因為運算子執行緒與運算子具有相同的名稱，並且會指出與哪個子任務相關聯。例如，以下堆疊追蹤來自 KeyedProcess 運算子，並且是第 1 個子任務。

```
"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(KeyedProcessOperat
    at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStr
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTas
    at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProces
    ...
```

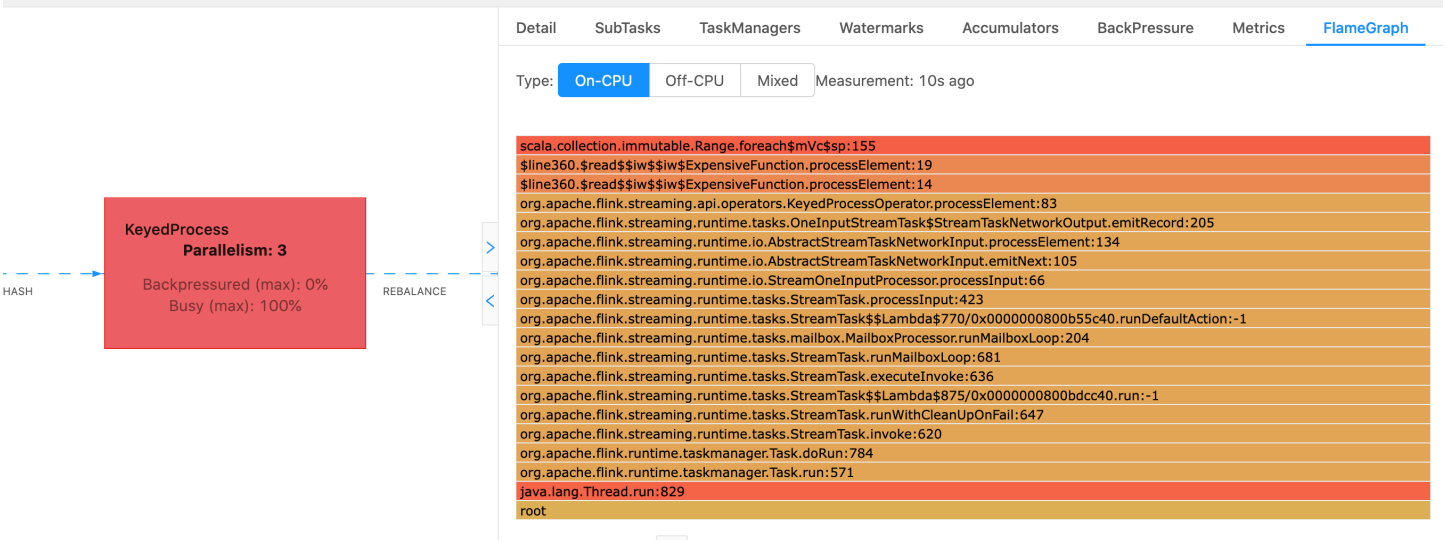
如果有多個運算子具有相同名稱，則可能會造成混淆，但我們可以透過命名運算子來解決這個問題。例如：

```
....
.process(new ExpensiveFunction).name("Expensive function")
```

火焰圖

火焰圖是一款有用的偵錯工具，它可以可視化目標程式碼的堆疊追蹤，從而允許識別最常見的程式碼路徑。它們透過對堆疊追蹤進行多次取樣來建立。火焰圖的 x 軸顯示不同的堆疊設定檔，y 軸顯示堆疊深度，以及堆疊追蹤中的呼叫。火焰圖中的單個矩形顯示在堆疊框架上，框架的寬度顯示它在堆疊中出現的頻率。如需火焰圖表及其用法的詳細資訊，請參閱火焰圖。

在 Flink 中，運算子的火焰圖可以透過 Web UI 存取，方法是選取運算子，然後選擇火焰圖標籤。一旦收集到足夠的樣本，火焰圖即會顯示。以下是花費了大量時間執行檢查點的 ProcessFunction 的火焰圖。



這是一個非常簡單的火焰圖，其中顯示了所有 CPU 時間都花費在一個 foreach 迴圈內的 ExpensiveFunction 運算子的 `processElement` 內。您還可以取得行號，以幫助確定程式碼的執行位置。

檢查點逾時

如果應用程式未最佳化或正確佈建，檢查點可能會失敗。本節說明此狀況的徵狀和疑難排解步驟。

徵狀

如果應用程式的檢查點失敗，`numberOfFailedCheckpoints` 將會大於零。

檢查點可能會因為直接失敗 (例如應用程式錯誤) 或暫時性失敗 (例如應用程式資源不足) 而失敗。檢查應用程式日誌和指標是否有下列徵狀：

- 程式碼中有錯誤。
- 存取應用程式相依服務時發生錯誤。

- 序列化資料時發生錯誤。如果預設的序列化程式無法序列化應用程式資料，則應用程式將失敗。如需在應用程式中使用自訂序列化程式的相關資訊，請參閱《Apache Flink 文件》https://nightlies.apache.org/flink/flink-docs-release-1.15/中的自訂序列化程式。
- 記憶體不足錯誤。
- 以下指標急劇增加或穩定增加：
  - `heapMemoryUtilization`
  - `oldGenerationGCTime`
  - `oldGenerationGCCount`
  - `lastCheckpointSize`
  - `lastCheckpointDuration`

如需監控檢查點的詳細資訊，請參閱https://nightlies.apache.org/flink/flink-docs-release-1.15/《Apache Flink 文件》中的監控檢查點。

原因與解決方案

您的應用程式日誌錯誤訊息會顯示直接失敗的原因。暫時性失敗可能有下列原因：

- 應用程式佈建的 KPU 不足。如需增加應用程式佈建的相關資訊，請參閱擴展。
- 應用程式狀態大小太大。您可以使用 `lastCheckpointSize` 指標監控應用程式狀態大小。
- 應用程式的狀態資料在索引鍵之間分配不平均。如果應用程式使用 KeyBy 運算子，請確保您的傳入資料在索引鍵之間已平均分割。如果將大部分資料指派給單一索引鍵，則會產生瓶頸，從而導致失敗。
- 應用程式遇到記憶體背壓或垃圾回收背壓。監控應用程式的 `heapMemoryUtilization`、`oldGenerationGCTime` 以及 `oldGenerationGCCount`，看是否有值在急劇增加或穩定增加。

Apache Beam 應用程式檢查點失敗

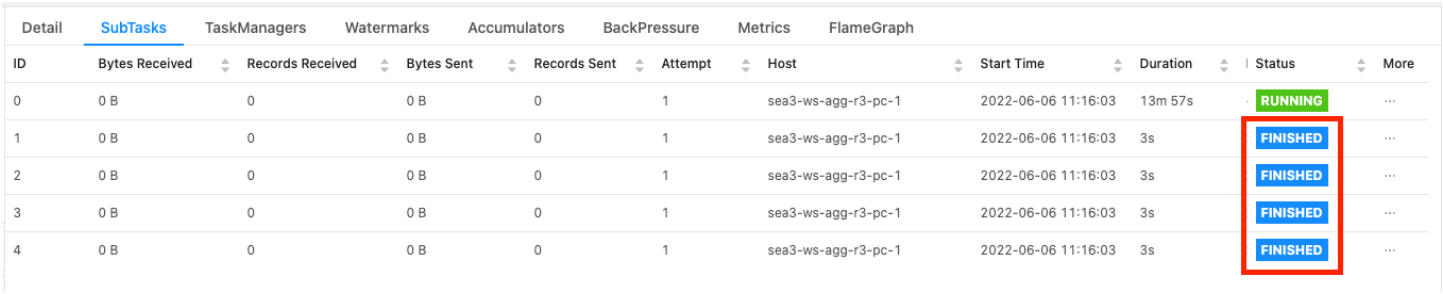
如果設定 Beam 應用程式時將 shutdownSourcesAfterIdleMs 設定為 0ms，則檢查點可能無法觸發，因為任務處於「FINISHED」狀態。本節說明此狀況的徵狀和解決方案。

徵狀

前往 Managed Service for Apache Flink 應用程式 CloudWatch 日誌，檢查其中是否記錄了下列日誌訊息。下列日誌訊息指出檢查點無法觸發，因為某些任務已完成。

```
                {
                "locationInformation":
"org.apache.flink.runtime.checkpoint.CheckpointCoordinator.onTriggerFailure(CheckpointCoordina
                "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"
                }
```

這也可以在 Flink 儀表板上找到，其中一些任務已進入「FINISHED」狀態，並且無法再執行檢查點。



原因

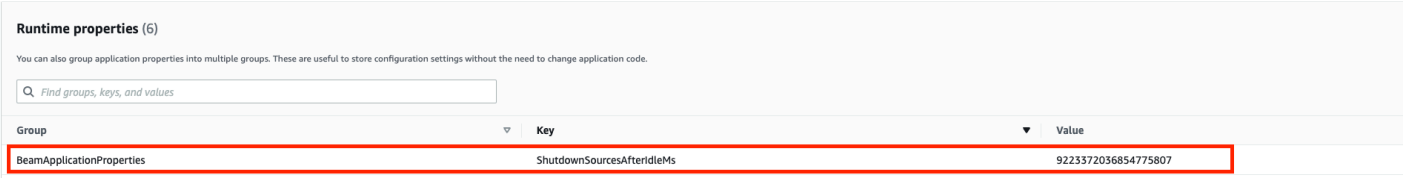
shutdownSourcesAfterIdleMs 是 Beam 組態變數，可關閉閒置了一段設定時間 (毫秒) 的來源。一旦來源關閉，無法再執行檢查點。這可能導致檢查點失敗。

任務進入「FINISHED」狀態的其中一個原因是當 shutdownSourcesAfterIdleMs 設定為 0ms 時，意味著閒置的任務將立即關閉。

解決方案

若要防止任務立即進入「FINISHED」狀態，請將 shutdownSourcesAfterIdleMs 設定為長 Long.MAX_VALUE。這可以透過兩種方式進行：

- 選項 1：如果 Beam 組態是在 Managed Service for Apache Flink 應用程式的組態頁面中設定，則可以新增一個索引鍵-值對來設定 shutdpwnSourcesAfteridleMs，如下所示：



- 選項 2：如果 Beam 組態是在 JAR 檔案中設定，您可以按如下方式設定 shutdownSourcesAfterIdleMs：

```
                        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
```

背壓

Flink 使用背壓來調整個別運算子的處理速度。

由於許多原因，運算子可能難以跟上處理收到的訊息量。該操作可能需要比運算子可用資源更多的 CPU 資源，運算子可能會等待 I/O 操作完成。如果運算子無法以足夠快的速度處理事件，它會在饋送給慢速運算子的上游運算子中造成背壓。這會導致上游運算子減慢速度，從而進一步將背壓傳播到來源，並透過減慢速度來使來源適應應用程式的整體輸送量。您可以在 Apache Flink™ 如何處理背壓中找到有關背壓及其運作方式的更深入說明。

知道應用程式中的哪些運算子速度緩慢，可為您提供重要資訊來了解應用程式效能問題的根本原因。背壓資訊透過 Flink 控制面板公開。若要識別慢速運算子，請尋找具有最接近接收器高背壓值的運算子 (以下範例中為運算子 B)。造成緩慢的運算子就是其中一個下游運算子 (範例中的運算子 C)。B 可以更快地處理事件，但由於無法將輸出轉發到實際的慢速運算子 C，因此會受到背壓。

```
A (backpressured 93%) -> B (backpressured 85%) -> C (backpressured 11%) -> D
 (backpressured 0%)
```

一旦確定慢速運算子，試著了解它為什麼慢。可能有無數的原因，有時出了什麼問題並不明顯，可能需要數天的偵錯和分析來解決。以下是一些明顯和更常見的原因，其中一些進一步解釋如下：

- 運算子正在執行緩慢的 I/O，例如網路呼叫 (考慮改用 AsynCIO)。
- 存在資料扭曲，一個運算子接收的事件比其他運算子多，可查看 Flink 儀表板中單個子任務 (即同一運算子的多個執行個體) 的進/出訊息數目進行驗證。
- 這是一個消耗資源的操作 (如果沒有資料扭曲，請考慮擴展 CPU/記憶體綁定的工作或增加I/O 綁定工作的 `ParallelismPerKPU`)
- 運算子中存在大量記錄 (將生產應用程式的記錄減少到最低限度，或者考慮將偵錯輸出發送到資料串流）。

使用丟棄的接收器測試輸送量

丟棄接收器只是忽略它在仍在執行應用程式時收到的所有事件 (沒有任何接收器的應用程式無法執行)。這對於輸送量測試、分析以及驗證應用程式是否正確擴展非常有用。這也是一種非常簡潔實用的完整性檢查，可以驗證接收器是否給應用程式導致背壓 (不過直接檢查背壓指標通常更容易和更直接)。

您可以透過用捨棄的接收器取代應用程式的所有接收器，並建立可產生與生產資料類似資料的模擬來源，來衡量應用程式在特定平行處理設定時的最大輸送量。然後，您也可以增加平行處理層級，以驗證應用程式是否正確擴展，並且沒有只在較高輸送量 (例如由於資料扭曲) 時才會出現的瓶頸。

資料扭曲

Flink 應用程式在叢集上分佈式執行。為了橫向擴展到多個節點，Flink 使用了鍵控串流的概念，這實質上意味著某個串流的事件將依據特定索引鍵 (例如客戶 ID) 進行分割，然後 Flink 可以處理不同節點上的不同分割區。許多 Flink 運算子然後會根據這些分割區評估，例如鍵控視窗、處理函數和非同步 I/O。

選擇分割區索引鍵通常取決於業務邏輯。同時，許多最佳實務 (例如 DynamoDB 和 Spark) 同樣適用於 Flink，包括：

- 確保分割區索引鍵的高基數
- 避免分割區之間的事件量扭曲

您可以比較 Flink 儀表板中接收/發送的子任務 (即同一運算子的多個執行個體) 的記錄數來識別分割區中是否存在扭曲。此外，Managed Service for Apache Flink 監控也可設定為公開子任務層級的 `numRecordsIn/Out` 和 `numRecordsInPerSecond/OutPerSecond` 指標。

狀態扭曲

對於有狀態運算子，即負責維護其業務邏輯 (如窗口) 狀態的運算子，資料扭曲總是會導致狀態扭曲。由於資料扭曲，某些子任務比其他子任務收到更多的事件，因此也在狀態中保留了更多資料。但是，即使對於具有均勻平衡分割區的應用程式，在狀態中保留多少資料也可能會出現扭曲。例如，對於工作階段窗口，某些使用者和工作階段分別都可能比其他使用者和工作階段長得多。如果較長的工作階段恰好是相同分割區的一部分，則可能導致相同運算子的不同子任務所保留的狀態大小不平衡。

狀態扭曲不僅增加了個別子任務所需的記憶體和磁碟資源，還會降低應用程式的整體效能。當應用程式取得檢查點或儲存點時，運算子狀態會保留在 Amazon S3 種，以保護狀態免受節點或叢集故障影響。在此處理程序期間 (特別是在 Managed Service for Apache Flink 上預設只啟用恰好一次的語義中)，處理會從外部暫停，直到檢查點/儲存點執行完成為止。如果有資料扭曲，則完成操作的時間可能會受到已累積了特別大量的狀態之單一子任務所限制。在極端情況下，由於單個子任務無法持續保留狀態，擷取檢查點/儲存點可能會失敗。

因此，與資料扭曲類似，狀態扭曲也會大幅降低應用程式執行速度。

若要識別狀態扭曲，可以利用 Flink 儀表板。尋找最新的檢查點或儲存點，並在詳細資料中比較已針對個別子任務儲存的資料量。

與不同地區的資源整合

您可以啟用 `StreamingFileSink`，透過 Flink 組態中跨區域複寫所需的設定，從 Managed Service for Apache Flink 應用程式寫入不同區域中的 Amazon S3 儲存貯體。若要這樣做，請在 AWS Support 中心填寫支援票證。

Amazon Managed Service for Apache Flink 文件歷史記錄

下表說明自 Managed Service for Apache Flink 推出上一個版本以來後，文件內所進行的重要變更。

- API 版本：2018-05-23
- 文件最新更新時間：2023 年 8 月 30 日

變更	描述	日期
Kinesis Data Analytics 現在稱為 Managed Service for Apache Flink	服務端點、API、命令列介面、IAM 存取政策、指 CloudWatch 標或AWS帳單儀表板沒有變更。現有的應用程式將繼續像先前一樣運作。如需詳細資訊，請參閱什麼是 Managed Service for Apache Flink？	2023 年 8 月 30 日
支援 Apache Flink 1.15.2 版	Managed Service for Apache Flink 現支援使用 Apache Flink 1.15.2 版的應用程式。使用 Apache Flink 資料表 API 建立 Kinesis Data Analytics 應用程式。如需詳細資訊，請參閱建立應用程式。	2022 年 11 月 22 日
支援 Apache Flink 1.13.2 版	Managed Service for Apache Flink 現支援使用 Apache Flink 1.13.2 版的應用程式。使用 Apache Flink 資料表 API 建立 Kinesis Data Analytics 應用程式。如需詳細資訊，請參閱 Flink 1.13.2 入門。	2021 年 10 月 13 日
支援 Python	Managed Service for Apache Flink 現支援搭配使用 Python	2021 年 3 月 25 日

變更	描述	日期
	與 Apache Flink 資料表 API & SQL 的應用程式。如需詳細資訊，請參閱 使用 Python。	
支援 Apache Flink 1.11.1	Managed Service for Apache Flink 現支援使用 Apache Flink 1.11.1 版的應用程式。使用 Apache Flink 資料表 API 建立 Kinesis Data Analytics 應用程式。如需詳細資訊，請參閱 建立應用程式。	2020 年 11 月 19 日
Apache Flink 儀表板	使用 Apache Flink 儀表板來監控應用程式運作狀態和效能。如需詳細資訊，請參閱 Apache Flink 儀表板。	2020 年 11 月 19 日
增強型扇出 (EFO) 取用者	建立使用增強型扇出 (EFO) 取用者從 Kinesis 資料串流讀取的應用程式。如需詳細資訊，請參閱 增強型扇出 (EFO) 取用者。	2020 年 10 月 6 日
Apache Beam	建立使用 Apache Beam 處理串流資料的應用程式。如需詳細資訊，請參閱 搭配使用 CloudFormation 與 Managed Service for Apache Flink。	2020 年 9 月 15 日
效能	如何疑難排解應用程式效能問題，以及如何建立高效能的應用程式。如需詳細資訊，請參閱 效能。	2020 年 7 月 21 日

變更	描述	日期
自訂金鑰存放區	如何存取使用自訂金鑰存放區進行傳輸加密的 Amazon MSK 叢集。如需詳細資訊，請參閱自訂信任存放區。	2020 年 6 月 10 日
CloudWatch 警報	使用適用於 Apache Flink 的受管理服務建立 CloudWatch 警示的建議。如需詳細資訊，請參閱 警示。	2020 年 6 月 5 日
新 CloudWatch 量度	Apache Flink 的受管服務現在會向 Amazon CloudWatch 指標發出 22 個指標。如需詳細資訊，請參閱 Managed Service for Apache Flink 中的指標和維度。	2020 年 5 月 12 日
自訂 CloudWatch 指標	定義應用程式特定的指標，並將其發送到 Amazon CloudWatch 指標。如需詳細資訊，請參閱 自訂指標。	2020 年 5 月 12 日
範例：從不同帳戶的 Kinesis 串流中讀取	了解如何在 Managed Service for Apache Flink 應用程式中，使用不同 AWS 帳戶存取 Kinesis 串流。如需詳細資訊，請參閱 跨帳戶。	2020 年 3 月 30 日
支援 Apache Flink 1.8.2	Managed Service for Apache Flink 現支援使用 Apache Flink 1.8.2 的應用程式。使用 Flink StreamingFileSink 連接器將輸出直接寫入 S3。如需詳細資訊，請參閱 建立應用程式。	2019 年 12 月 17 日

變更	描述	日期
Managed Service for Apache Flink VPC	設定 Managed Service for Apache Flink 應用程式連線至虛擬私有雲端 (VPC)。如需詳細資訊，請參閱 使用 Amazon VPC。	2019 年 11 月 25 日
Managed Service for Apache Flink 最佳實務	建立和管理 Managed Service for Apache Flink 應用程式的最佳做法。如需詳細資訊，請參閱 最佳實務。	2019 年 10 月 14 日
分析 Managed Service for Apache Flink 應用程式日誌	使用 CloudWatch 日誌深入解析來監視 Apache Flink 應用程式的受管理服務。如需詳細資訊，請參閱 分析日誌。	2019 年 6 月 26 日
Managed Service for Apache Flink 應用程式執行時間屬性	使用 Managed Service for Apache Flink 中的執行時間屬性。如需詳細資訊，請參閱執行時間屬性。	2019 年 6 月 24 日
標記 Managed Service for Apache Flink 應用程式	使用應用程式標記來判斷每個應用程式的成本、控制存取或用於使用者定義之目的。如需詳細資訊，請參閱使用標籤。	2019 年 5 月 8 日
Managed Service for Apache Flink 範例應用程式	適用於 Apache Flink 的受管理服務應用程式範例，示範視窗操作員並將 CloudWatch 輸出寫入記錄檔。如需詳細資訊，請參閱 範例。	2019 年 5 月 1 日

變更	描述	日期
使用 AWS CloudTrail 記錄 Managed Service for Apache Flink API 呼叫	Managed Service for Apache Flink 已與 AWS CloudTrail 整合，這是一項服務，可提供使用者、角色或 AWS 服務在 Managed Service for Apache Flink 中所採取動作的記錄。如需詳細資訊，請參閱使用 AWS CloudTrail。	2019 年 3 月 22 日
建立應用程式 (Kinesis Data Firehose Sink)	將 Amazon Kinesis 資料串流作為來源，將 Amazon Kinesis Data Firehose 串流作為接收器，藉此練習建立 Managed Service for Apache Flink。如需詳細資訊，請參閱Kinesis Data Firehose 接收器。	2018 年 12 月 13 日
公開發行	這是《適用於 Java 應用程式的 Managed Service for Apache Flink 開發人員指南》的初始版本。	2018 年 11 月 27 日

Managed Service for Apache Flink API 範例程式碼

本主題包含適用於 Managed Service for Apache Flink 動作的範例請求區塊。

若要使用 JSON 作為 AWS Command Line Interface (AWS CLI) 動作的輸入，請將请求儲存在 JSON 檔案中。然後使用 `--cli-input-json` 參數將檔案名稱傳遞至動作。

以下範例示範如何將 JSON 檔案用於動作中。

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

如需關於搭配使用 JSON 與 AWS CLI 之詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的產生 CLI 骨架和 CLI 輸入 JSON 參數。

主題

- AddApplicationCloudWatchLoggingOption
- AddApplicationInput
- AddApplicationInputProcessingConfiguration
- AddApplicationOutput
- AddApplicationReferenceDataSource
- AddApplicationVpcConfiguration
- CreateApplication
- CreateApplicationSnapshot
- DeleteApplication
- DeleteApplicationCloudWatchLoggingOption
- DeleteApplicationInputProcessingConfiguration
- DeleteApplicationOutput
- DeleteApplicationReferenceDataSource
- DeleteApplicationSnapshot
- DeleteApplicationVpcConfiguration
- DescribeApplication
- DescribeApplicationSnapshot

- DiscoverInputSchema
- ListApplications
- ListApplicationSnapshots
- StartApplication
- StopApplication
- UpdateApplication

AddApplicationCloudWatchLoggingOption

AddApplicationCloudWatchLoggingOption 動作的下列範例請求程式碼會將 Amazon CloudWatch 日誌選項新增至 Managed Service for Apache Flink 應用程式：

```
{
    "ApplicationName": "MyApplication",
    "CloudWatchLoggingOption": {
        "LogStreamARN": "arn:aws:logs:us-east-1:123456789123:log-group:my-log-
group:log-stream:My-LogStream"
    },
    "CurrentApplicationVersionId": 2
}
```

AddApplicationInput

AddApplicationInput 動作的下列範例請求程式碼會將應用程式輸入新增至 Managed Service for Apache Flink 應用程式：

```
{
   "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

AddApplicationInputProcessingConfiguration 動作的下列範例請求程式碼會將應用程式輸入處理組態新增至 Managed Service for Apache Flink 應用程式：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "InputId": "2.1",
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "arn:aws:lambda:us-
east-1:012345678901:function:MyLambdaFunction"
      }
   }
}
```

AddApplicationOutput

AddApplicationOutput 動作的下列範例請求程式碼會將 Kinesis 資料串流作為應用程式輸出新增至 Managed Service for Apache Flink 應用程式：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "Output": {
      "DestinationSchema": {
         "RecordFormatType": "JSON"
      },
      "KinesisStreamsOutput": {
         "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleOutputStream"
      },
      "Name": "DESTINATION_SQL_STREAM"
   }
}
```

AddApplicationReferenceDataSource

AddApplicationReferenceDataSource 動作的下列範例請求程式碼會將 CSV 應用程式參考資料來源新增至 Managed Service for Apache Flink 應用程式：

```
{
   "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 動作的下列範例請求程式碼會將 VPC 組態新增至現有的應用程式：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
   }
}
```

CreateApplication

CreateApplicatio 動作的下列範例請求程式碼會建立 Managed Service for Apache Flink 應用程式：

```
{
  "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

CreateApplicationSnapshot 動作的下列範例請求程式碼會建立應用程式狀態的快照：

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

DeleteApplication

DeleteApplication 動作的下列範例請求程式碼會刪除 Managed Service for Apache Flink 應用程式：

```
{"ApplicationName": "MyApplication",
"CreateTimestamp": 12345678912}
```

DeleteApplicationCloudWatchLoggingOption

DeleteApplicationCloudWatchLoggingOption 動作的下列範例請求程式碼會從 Managed Service for Apache Flink 應用程式中刪除 Amazon CloudWatch 記錄選項：

```
{
    "ApplicationName": "MyApplication",
    "CloudWatchLoggingOptionId": "3.1"
    "CurrentApplicationVersionId": 3
}
```

DeleteApplicationInputProcessingConfiguration

DeleteApplicationInputProcessingConfiguration 動作的下列範例請求程式碼會從 Managed Service for Apache Flink 應用程式中移除輸入處理組態：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "InputId": "2.1"
}
```

DeleteApplicationOutput

DeleteApplicationOutput 動作的下列範例請求程式碼會從 Managed Service for Apache Flink 應用程式中移除應用程式輸出：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "OutputId": "4.1"
}
```

DeleteApplicationReferenceDataSource

DeleteApplicationReferenceDataSource 動作的下列範例請求程式碼會從 Managed Service for Apache Flink 應用程式中移除應用程式參考資料來源：

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5,
   "ReferenceId": "5.1"
}
```

DeleteApplicationSnapshot

DeleteApplicationSnapshot 動作的下列範例請求程式碼會刪除應用程式狀態的快照：

```
{
   "ApplicationName": "MyApplication",
   "SnapshotCreationTimestamp": 12345678912,
   "SnapshotName": "MySnapshot"
}
```

DeleteApplicationVpcConfiguration

DeleteApplicationVpcConfiguration 動作的下列範例請求程式碼會從應用程式中移除現有的 VPC 組態：

```
{
```

```
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfigurationId": "1.1"
}
```

DescribeApplication

DescribeApplication 動作的下列範例請求程式碼會傳回 Managed Service for Apache Flink 應用程式的詳細資訊：

```
{"ApplicationName": "MyApplication"}
```

DescribeApplicationSnapshot

DescribeApplicationSnapshot 動作的下列請求程式碼範例會傳回應用程式狀態快照的詳細資訊：

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

DiscoverInputSchema

DiscoverInputSchema 動作的下列請求程式碼範例會從串流來源產生結構描述：

```
{
   "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"
}
```

DiscoverInputSchema 動作的下列請求程式碼範例會從參考來源產生結構描述：

```
{
   "S3Configuration": {
      "BucketARN": "arn:aws:s3:::mybucket",
      "FileKey": "TickerReference.csv"
   },
   "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

ListApplications

ListApplications 動作的下列範例請求程式碼會傳回您帳戶中 Managed Service for Apache Flink 應用程式的清單：

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

ListApplicationSnapshots

ListApplicationSnapshots 動作的下列請求程式碼範例會傳回應用程式狀態的快照清單：

```
{"ApplicationName": "MyApplication",
   "Limit": 50,
   "NextToken": "aBcDeFgHiJkLmNoPqRsTuVwXyZ0123"
}
```

StartApplication

StartApplication 動作的下列範例請求程式碼會啟動 Managed Service for Apache Flink 應用程式，並從最新的快照 (如有) 載入應用程式狀態：

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

StopApplication

API_StopApplication 動作的下列範例請求程式碼會停止 Managed Service for Apache Flink 應用程式：

```
{"ApplicationName": "MyApplication"}
```

UpdateApplication

UpdateApplication 動作的下列範例請求程式碼會更新 Managed Service for Apache Flink 應用程式，以變更應用程式程式碼的位置：

```
{"ApplicationName": "MyApplication",
"CurrentApplicationVersionId": 1,
"ApplicationConfigurationUpdate": {
      "ApplicationCodeConfigurationUpdate": {
         "CodeContentTypeUpdate": "ZIPFILE",
         "CodeContentUpdate": {
            "S3ContentLocationUpdate": {
               "BucketARNUpdate": "arn:aws:s3:::my_new_bucket",
               "FileKeyUpdate": "my_new_code.zip",
               "ObjectVersionUpdate": "2"
         }
      }
   }
}
```

Managed Service for Apache Flink API 參考

如需 Managed Service for Apache Flink 所提供之 API 的相關資訊，請參閱 Managed Service for Apache Flink API 參考。