



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

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

# Table of Contents

|  |     |
|--|-----|
| .....  | xvi |
| 什麼是阿帕奇 Flink 的託管服務？ .....                            | 1   |
| 決定使用阿帕奇 Flink 的管理服務或管理服務為阿帕奇 Flink 工作室 .....         | 1   |
| 選擇要在 Apache Flink APIs 的受管理服務中使用的 Apache Flink ..... | 3   |
| 選擇一個快速連結 API .....                                   | 3   |
| 開始使用串流資料應用程式 .....                                   | 4   |
| 運作方式 .....   | 6   |
| 編程您的阿帕奇 Flink 應用程式 .....                             | 6   |
| DataStream API .....                                 | 6   |
| 表 API .....  | 7   |
| 為 Apache Flink 應用程式建立您的受管理服務 .....                   | 7   |
| 為 Apache Flink 應用程式建立受管理服務 .....                     | 8   |
| 建立適用於 Apache Flink 應用程式程式碼的託管服務 .....                | 8   |
| 為 Apache Flink 應用程式建立您的受管理服務 .....                   | 9   |
| 啟動適用於 Apache Flink 應用程式的管理服務 .....                   | 10  |
| 驗證您的 Apache Flink 應用程式託管服務 .....                     | 10  |
| 為您的 Apache Flink 應用程式的受管理服務啟用系統復原 .....              | 11  |
| 為阿帕奇 Flink 應用程式運行託管服務 .....                          | 13  |
| 識別申請和工作狀態 .....                                      | 13  |
| 執行批次工作量 .....  | 14  |
| 檢閱 Apache Flink 應用程式資源的受管理服務 .....                   | 14  |
| 阿帕奇 Flink 應用程式資源的管理服務 .....                          | 15  |
| 阿帕奇 Flink 應用程式資源 .....                               | 15  |
| 檢閱 DataStream API 元件 .....                           | 16  |
| 使用連接器在適用於 Apache Flink 的受管理服務中移動資料 .....             | 16  |
| 在 Apache Flink 的受管理服務中使用運算子轉換資料 .....                | 25  |
| 在適用於 Apache Flink 的受管理服務中追蹤事件 .....                  | 26  |
| 檢閱表格 API 元件 .....                                    | 27  |
| 使用表格 API 連接器 .....                                   | 27  |
| 檢閱表格 API 時間屬性 .....                                  | 28  |
| 使用 Python 與阿帕奇 Flink 的託管服務 .....                     | 29  |
| 為阿帕奇 Flink Python 應用程式編程您的託管服務 .....                 | 29  |
| 為阿帕奇 Flink Python 應用程式創建您的託管服務 .....                 | 32  |
| 監視您的阿帕奇 Flink Python 應用程式的託管服務 .....                 | 33  |

|  |    |
|--|----|
| 在 Apache Flink 的受管理服務中使用執行階段屬性 .....             | 35 |
| 使用主控台管理執行階段屬性 .....                              | 35 |
| 使用管理執行階段屬性 CLI .....                             | 36 |
| 在 Apache Flink 應用程式的受管理服務中存取執行階段屬性 .....         | 38 |
| 搭配 Apache Flink 的受管理服務使用 Apache Flink 連接器 .....  | 39 |
| 在 Apache Flink 的受管理服務中實作容錯能力 .....               | 41 |
| 在 Apache Flink 的受管理服務中設定檢查點 .....                | 41 |
| 檢閱檢查點範例 API .....                                | 42 |
| 使用快照管理應用程式備 .....                                | 44 |
| 管理自動建立快照 .....                                   | 45 |
| 從包含不相容狀態資料的快照還原 .....                            | 46 |
| 檢閱快照API範例 .....                                  | 47 |
| 為 Apache Flink 使用就地版本升級 .....                    | 49 |
| 使用 Apache Flink 的就地版本升級來升級應用程式 .....             | 50 |
| 將您的應用程式升級到新的 Apache Flink 版本 .....               | 51 |
| 還原應用程式升級 .....                                   | 56 |
| 應用程式升級的一般最佳作法和建議 .....                           | 56 |
| 應用程式升級時的注意事項和已知 .....                            | 57 |
| 在 Apache Flink 的受管理服務中實作應用程式調整 .....             | 58 |
| 設定應用程式平行處理和 ParallelismPer KPU .....             | 58 |
| 分配 Kinesis 處理單元 .....                            | 59 |
| 更新應用程式的平行處理 .....                                | 60 |
| 在適用於 Apache Flink 的受管理服務中使用自動調整 .....            | 61 |
| maxParallelism 考量 .....                          | 63 |
| 將標籤新增至 Apache Flink 應用程式的受管理服務 .....             | 64 |
| 建立應用程式時新增標籤 .....                                | 64 |
| 新增或更新現有應用程式的標籤 .....                             | 65 |
| 列出應用程式的標籤 .....                                  | 65 |
| 從應用程式移除標籤 .....                                  | 66 |
| 搭 CloudFormation 配 Apache Flink 的受管理服務搭配使用 ..... | 66 |
| 開始之前 .....                                       | 66 |
| 編寫一個 Lambda 函數 .....                             | 66 |
| 建立 Lambda 角色 .....                               | 68 |
| 叫用 Lambda 函數 .....                               | 69 |
| 檢閱延伸範例 .....                                     | 69 |
| 搭配 Apache Flink 管理服務使用 Apache Flink 儀表板 .....    | 75 |

|   |     |
|---|-----|
| 存取應用程式的 Apache Flink 儀表板 .....  | 75  |
| 發行版本 .....  | 77  |
| 阿帕奇 Flink 1.19 Amazon 託管服務 .....  | 78  |
| 支援的功能 .....   | 78  |
| 阿帕奇 Flink 1.19.1 Amazon 託管服務的變化 .....                                     | 80  |
| 元件 .....  | 81  |
| 已知問題 .....  | 81  |
| 阿帕奇 FLINK 1.18 Amazon 託管服務 .....  | 81  |
| 使用 Apache Flink 1.15 的 Amazon Managed Service for Apache Flink 中的變更 ..... | 83  |
| 元件 .....  | 83  |
| 錯誤修正 .....  | 84  |
| 已知問題 .....  | 84  |
| 阿帕奇 Flink 1.15 的 Amazon 託管服務 .....  | 84  |
| 使用 Apache Flink 1.15 的 Amazon Managed Service for Apache Flink 中的變更 ..... | 83  |
| 元件 .....  | 83  |
| 早期版本 .....  | 87  |
| 使用阿帕奇 Flink Kinesis 串流連接器與先前的 Apache Flink 版本 .....                       | 88  |
| 使用阿帕奇 FLINK 1.8.2 建置應用程式 .....  | 89  |
| 使用阿帕奇 FLINK 1.6.2 建置應用程式 .....  | 89  |
| 升級應用 .....  | 90  |
| 阿帕奇 FLINK 1.6.2 和 1.8.2 中的可用連接器 .....                                     | 91  |
| Flink 1.13.2 入門 .....   | 91  |
| Flink 1.11.1 入門 .....   | 115 |
| 開始使用：Flink 1.8.2-棄用 .....   | 139 |
| 開始使用：Flink 1.6.2-棄用 .....   | 162 |
| 舊式範例 .....  | 185 |
| 搭配 Apache Flink 的受管理服務使用工作室筆記本 .....                                      | 346 |
| 使用正確的 Studio 筆記本運行時版本 .....   | 347 |
| 建立工作室筆記本 .....  | 347 |
| 對串流資料執行互動式分析 .....  | 348 |
| Flink 解譯器 .....   | 349 |
| Apache Flink 資料表環境變數 .....  | 350 |
| 部署為具有持久狀態的應用程式 .....  | 350 |
| Scala/Python 條件 .....   | 351 |
| SQL 標準 .....  | 352 |
| IAM 權限 .....  | 352 |

|   |     |
|---|-----|
| 使用連接器和相依性 .....   | 353 |
| 預設連接器 .....   | 353 |
| 新增相依性和自訂連接器 .....   | 354 |
| 使用者定義的函數 .....  | 355 |
| 使用使用者定義函數的考量 .....  | 356 |
| 啟用檢查點 .....   | 357 |
| 設定檢查點間隔 .....   | 357 |
| 設定檢查點類型 .....   | 358 |
| 升級工作室運行 .....   | 358 |
| 將您的筆記本升級到新的 Studio 運行時 .....                              | 358 |
| 使用 AWS Glue .....   | 363 |
| 資料表屬性 .....   | 363 |
| 適用於 Apache Flink 管理服務中的工作室筆記本的範例和教學課程 .....               | 365 |
| 教學課程：在 Apache Flink 的受管理服務中建立工作室筆記本 .....                 | 365 |
| 教學課程：將 Studio 筆記本部署為具有持久狀態的 Apache Flink 應用程式的受管理服務 ..... | 383 |
| 查看示例查詢以分析 Studio 筆記本中的數據 .....                            | 386 |
| 疑難排解適用於 Apache Flink 的受管理服務的工作室 .....                     | 398 |
| 停止卡住的應用程式 .....   | 398 |
| 在沒有網際網路存取的情況下部署為VPC具有持久狀態的應用程式 .....                      | 398 |
| Deploy-as-app 尺寸和構建時間縮短 .....                             | 399 |
| 取消工作 .....  | 401 |
| 重新啟動阿帕奇 Flink 解釋器 .....                                   | 402 |
| 為 Apache Flink 工作室筆記本的受管理服務建立自訂IAM原則 .....                | 402 |
| AWS Glue .....  | 402 |
| CloudWatch 日誌 .....                                       | 403 |
| Kinesis 串流 .....  | 404 |
| Amazon MSK 集群 .....                                       | 406 |
| 教學課程：開始使用適用於 Apache Flink 的 DataStream API受管理服務 .....     | 407 |
| 檢閱應用程式元 .....   | 140 |
| 完成必要的先決條件 .....   | 408 |
| 設定 帳戶 .....   | 409 |
| 註冊一個 AWS 帳戶 .....   | 92  |
| 建立具有管理存取權的使用者 .....                                       | 93  |
| 授與程式設計存取權 .....   | 411 |
| 後續步驟 .....  | 412 |
| 設定 AWS CLI .....  | 412 |

|   |     |
|---|-----|
| 下一步驟 .....                                  | 413 |
| 建立應用程式 .....                                | 413 |
| 建立相依資源 .....                                | 414 |
| 設定您的本機開發環境 .....                            | 416 |
| 下載並檢查阿帕奇 Flink 流 Java 代碼 .....              | 416 |
| 將樣本記錄寫入輸入流 .....                            | 420 |
| 在本機執行應用程式 .....                             | 422 |
| 觀察 Kinesis 串流中的輸入和輸出資料 .....                | 425 |
| 停止應用程式在本機執行 .....                           | 425 |
| 編譯並封裝應用程式程式碼 .....                          | 425 |
| 上傳應用程式代碼JAR文件 .....                         | 426 |
| 建立並設定 Apache Flink 應用程式的受管理服務 .....         | 426 |
| 下一步驟 .....                                  | 433 |
| 清除資源 .....                                  | 433 |
| 刪除適用於 Apache Flink 應用程式的受管理服務 .....         | 433 |
| 刪除您的 Kinesis 資料串流 .....                     | 434 |
| 刪除您的 Amazon S3 對象和存儲桶 .....                 | 434 |
| 刪除您的IAM資源 .....                             | 435 |
| 刪除您的 CloudWatch 資源 .....                    | 435 |
| 探索阿帕奇 Flink 的其他資源 .....                     | 435 |
| 探索其他資源 .....                                | 435 |
| 教學課程：開始在 Apache Flink 的受管理服務API中使用資料表 ..... | 437 |
| 檢閱應用程式元 .....                               | 437 |
| 完成必要的先決條件 .....                             | 438 |
| 建立應用程式 .....                                | 438 |
| 建立相依資源 .....                                | 439 |
| 設定您的本機開發環境 .....                            | 440 |
| 下載並檢查阿帕奇 Flink 流 Java 代碼 .....              | 440 |
| 在本機執行應用程式 .....                             | 446 |
| 觀察將資料寫入 S3 儲存貯體的應用程式 .....                  | 448 |
| 停止應用程式在本機執行 .....                           | 449 |
| 編譯並封裝應用程式程式碼 .....                          | 449 |
| 上傳應用程式碼JAR檔案 .....                          | 449 |
| 建立並設定 Apache Flink 應用程式的受管理服務 .....         | 450 |
| 下一步驟 .....                                  | 456 |
| 清除資源 .....                                  | 456 |

|  |     |
|--|-----|
| 刪除適用於 Apache Flink 應用程式的受管理服務 .....          | 456 |
| 刪除您的 Amazon S3 對象和存儲桶 .....                  | 456 |
| 刪除您的IAM資源 .....                              | 457 |
| 刪除您的 CloudWatch 資源 .....                     | 457 |
| 下一步驟 .....                                   | 458 |
| 探索其他資源 .....                                 | 458 |
| 教學課程：開始在 Apache Flink 的受管理服務中使用 Python ..... | 459 |
| 檢閱應用程式元 .....                                | 459 |
| 符合先決條件 .....                                 | 460 |
| 建立應用程式 .....                                 | 461 |
| 建立相依資源 .....                                 | 462 |
| 設定您的本機開發環境 .....                             | 463 |
| 下載並檢查阿帕奇 Flink 流 Python 代碼 .....             | 464 |
| 管理JAR相依性 .....                               | 467 |
| 將樣本記錄寫入輸入流 .....                             | 469 |
| 在本機執行應用程式 .....                              | 470 |
| 觀察 Kinesis 串流中的輸入和輸出資料 .....                 | 472 |
| 停止應用程式在本機執行 .....                            | 472 |
| Package 您的應用程式碼 .....                        | 473 |
| 將應用程式套件上傳到 Amazon S3 儲存貯體 .....              | 473 |
| 建立並設定 Apache Flink 應用程式的受管理服務 .....          | 473 |
| 下一步驟 .....                                   | 479 |
| 清除資源 .....                                   | 480 |
| 刪除適用於 Apache Flink 應用程式的受管理服務 .....          | 480 |
| 刪除您的 Kinesis 資料串流 .....                      | 480 |
| 刪除您的 Amazon S3 對象和存儲桶 .....                  | 480 |
| 刪除您的IAM資源 .....                              | 481 |
| 刪除您的 CloudWatch 資源 .....                     | 481 |
| 教程：開始在 Apache Flink 的託管服務中使用斯卡拉 .....        | 483 |
| 建立相依資源 .....                                 | 483 |
| 將樣本記錄寫入輸入流 .....                             | 484 |
| 下載並檢查應用程式程式碼 .....                           | 486 |
| 編譯和上傳應用程式的程式碼 .....                          | 487 |
| 創建並運行應用程序（控制台） .....                         | 488 |
| 建立應用程式 .....                                 | 488 |
| 設定應用程式 .....                                 | 489 |



|   |     |
|---|-----|
| 編輯IAM策略 .....   | 490 |
| 執行應用程式 .....  | 492 |
| 停止應用程式 .....  | 492 |
| 創建並運行應用程序 ( CLI ) .....                                   | 492 |
| 建立許可政策 .....  | 492 |
| 建立IAM策略 .....   | 494 |
| 建立應用程式 .....  | 496 |
| 啟動應用程式 .....  | 497 |
| 停止應用程式 .....  | 341 |
| 新增記 CloudWatch 錄選項 .....                                  | 342 |
| 更新環境屬性 .....  | 342 |
| 更新應用程式的程式碼 .....  | 343 |
| 清理 AWS 資源 .....   | 500 |
| 刪除適用於 Apache Flink 應用程式的受管理服務 .....                       | 500 |
| 刪除您的 Kinesis 資料串流 .....                                   | 500 |
| 刪除您的 Amazon S3 對象和存儲桶 .....                               | 501 |
| 刪除您的IAM資源 .....   | 501 |
| 刪除您的 CloudWatch 資源 .....                                  | 501 |
| 為 Apache Flink 應用程式使用具有託管服務的 Apache 光束 .....              | 502 |
| 阿帕奇 Flink 運行器與 Apache Flink 管理服務的局限性 .....                | 502 |
| 阿帕奇梁功能與阿帕奇 Flink 管理服務 .....                               | 502 |
| 使用 Apache Beam 建立應用程式 .....                               | 503 |
| 建立相依資源 .....  | 503 |
| 將樣本記錄寫入輸入流 .....  | 504 |
| 下載並檢查應用程式程式碼 .....  | 504 |
| 編譯應用程式程式碼 .....   | 506 |
| 上傳阿帕奇 Flink 流 Java 代碼 .....                               | 506 |
| 建立並執行 Apache Flink 應用程式的受管理服務 .....                       | 507 |
| 清除 .....  | 510 |
| 後續步驟 .....  | 512 |
| 培訓研討會、實驗室和解決方案實作 .....                                    | 513 |
| 阿帕奇 Flink 管理服務工作坊 .....                                   | 513 |
| 在部署至 Apache Flink 的受管理服務之前，先在本機開發 Apache Flink 應用程式 ..... | 513 |
| 使用 Managed Service for Apache Flink Studio 進行事件偵測 .....   | 514 |
| AWS 串流資料方案 .....  | 514 |
| 練習使用點擊流實驗室與阿帕奇 Flink 和阿帕奇卡夫卡 .....                        | 514 |

|   |     |
|---|-----|
| 使用 Application Auto Scaling 放設定自訂縮放 .....                     | 514 |
| 查看示例 Amazon CloudWatch 儀表板 .....                              | 515 |
| 使用適用於 Amazon 的 AWS 串流資料解決方案範本 MSK .....                       | 515 |
| 探索更多適用於 Apache Flink 解決方案的託管服務 GitHub .....                   | 515 |
| 為 Apache Flink 管理服務使用實用的公用程式 .....                            | 516 |
| 快照管理員 .....   | 516 |
| 基準測試 .....  | 516 |
| 為 Apache Flink 應用程式建立及使用受管理服務的範例 .....                        | 517 |
| 適用於阿帕奇 Flink 管理服務的 Java 範例 .....                              | 517 |
| 阿帕奇 Flink 管理服務的 Python 示例 .....                               | 519 |
| .....   | 519 |
| 斯卡拉示例為阿帕奇 Flink 管理服務 .....                                    | 520 |
| 阿帕奇 Flink 管理服務中的安全性 .....                                     | 522 |
| 適用於 Apache Flink 的受管理服務中的資料保護 .....                           | 522 |
| 資料加密 .....  | 522 |
| 適用於 Apache Flink 的受管理服務的 Identity and Access Management ..... | 523 |
| 物件 .....  | 524 |
| 使用身分驗證 .....  | 524 |
| 使用政策管理存取權 .....   | 527 |
| 阿帕奇 Flink 的 Amazon 託管服務如何與 IAM .....                          | 529 |
| 身分型政策範例 .....   | 535 |
| 故障診斷 .....  | 537 |
| 預防跨服務混淆代理人 .....  | 539 |
| 適用於 Apache Flink 的受管理服務的合規性驗證 .....                           | 540 |
| 美聯儲 RAMP .....  | 541 |
| 適用於 Apache Flink 的受管理服務中的彈性和災難復原 .....                        | 541 |
| 災難復原 .....  | 541 |
| 版本控制 .....  | 542 |
| Apache Flink 受管理服務中的基礎架構安全性 .....                             | 542 |
| 適用於 Apache Flink 的受管理服務的安全性最佳做法 .....                         | 542 |
| 實作最低權限存取 .....  | 543 |
| 使用IAM角色存取其他 Amazon 服務 .....                                   | 543 |
| 在相依資源實作伺服器端加密 .....   | 543 |
| 用 CloudTrail 於監控API通話 .....                                   | 543 |
| 在 Amazon 管理服務中為 Apache Flink 進行日誌記錄和監控 .....                  | 544 |
| 登入管理服務 .....  | 544 |

|  |     |
|--|-----|
| 使用記錄檔見解查詢 CloudWatch 記錄 .....                        | 545 |
| 在阿帕奇 Flink 的受管理服務中監視 .....                           | 545 |
| 在 Apache Flink 的受管理服務中設定應用程式登入 .....                 | 546 |
| 使用主控台設定 CloudWatch 記錄 .....                          | 547 |
| 使用設定 CloudWatch 記錄 CLI .....                         | 547 |
| 控制應用程式監控層 .....                                      | 552 |
| 套用記錄最佳做法 .....                                       | 552 |
| 執行記錄疑難排 .....  | 553 |
| 使用 CloudWatch 日誌見解 .....                             | 553 |
| 使用日誌見解分析 CloudWatch 日誌 .....                         | 553 |
| 執行範例查詢 .....   | 553 |
| 檢閱範例查詢 .....   | 554 |
| 適用於 Apache Flink 的受管理服務中的度量和維度 .....                 | 557 |
| 應用程式指標 .....   | 557 |
| Kinesis Data Streams 連接器測量結果 .....                   | 575 |
| Amazon MSK 連接器指標 .....                               | 576 |
| 阿帕奇齐柏林飞艇指标 .....                                     | 577 |
| 檢視 CloudWatch 指標 .....                               | 578 |
| 設定 CloudWatch 量度報告層級 .....                           | 578 |
| 搭配適用於 Apache Flink 的 Amazon 受管服務使用自訂指標 .....         | 580 |
| 搭配 Amazon 管理服務搭配 Apache Flink 使用 CloudWatch 警示 ..... | 583 |
| 將自訂訊息寫入 CloudWatch 記錄 .....                          | 590 |
| 使用 Log4j 寫入 CloudWatch 日誌 .....                      | 591 |
| 使用寫入 CloudWatch 記錄 SLF4J .....                       | 592 |
| 日誌管理服務的阿帕奇 Flink API 呼叫 AWS CloudTrail .....         | 593 |
| 中的 Apache 快速連結資訊的管理服務 CloudTrail .....               | 593 |
| 瞭解 Apache Flink 記錄檔項目的受管理服務 .....                    | 594 |
| 調整阿帕奇 Flink 的 Amazon 託管服務中的性能 .....                  | 597 |
| 解決效能問題 .....   | 597 |
| 瞭解資料路徑 .....   | 597 |
| 效能疑難排解 .....   | 597 |
| 使用效能最佳做法 .....                                       | 599 |
| 適當管理擴展 .....   | 600 |
| 監控外部相依性資源使用量 .....                                   | 601 |
| 在本機執行 Apache Flink 應用程式 .....                        | 602 |
| 監控效能 .....   | 602 |

|   |     |
|---|-----|
| 使用指標監視 CloudWatch 效能 .....                        | 602 |
| 使用 CloudWatch 記錄和警示來監控效能 .....                    | 602 |
| 適用於 Apache Flink 和工作室筆記本配額的託管服務 .....             | 603 |
| 管理適用於 Apache Flink 的受管理服務的維護工作 .....              | 605 |
| UUID為所有運營商設置一個 .....                              | 607 |
| 識別維護實例 .....                                      | 607 |
| 為您的 Apache Flink 應用程式的受管理服務實現生產準備 .....           | 608 |
| 負載測試您的應用程式 .....                                  | 608 |
| 定義最大平行度 .....                                     | 608 |
| UUID為所有運營商設置一個 .....                              | 609 |
| 為 Apache Flink 應用程式維護受管理服務的最佳做法 .....             | 610 |
| 容錯能力：檢查點和儲存點 .....                                | 610 |
| 不受支援的連接器版本 .....                                  | 611 |
| 效能與平行處理層級 .....                                   | 611 |
| 設定每個運算子的平行處理層級 .....                              | 612 |
| 日誌 .....  | 612 |
| 編碼 .....  | 612 |
| 管理憑證 .....  | 613 |
| 從具有較少碎片/分割區的來源讀取 .....                            | 613 |
| Studio 筆記本重新整理間隔 .....                            | 614 |
| Studio 筆記本最佳效能 .....                              | 614 |
| 浮水印策略和閒置碎片如何影響時間範圍 .....                          | 614 |
| Summary .....                                     | 615 |
| 範例 .....  | 616 |
| UUID為所有運營商設置一個 .....                              | 625 |
| 添加 ServiceResourceTransformer 到 Maven 的陰影插件 ..... | 625 |
| 阿帕奇 Flink 有狀態函數 .....                             | 627 |
| 阿帕奇 Flink 應用程序模板 .....                            | 627 |
| 模組組態的位置 .....                                     | 628 |
| 瞭解阿帕奇快顯設定 .....                                   | 629 |
| 阿帕奇 Flink 配置 .....                                | 629 |
| 狀態後端 .....  | 629 |
| 檢查點 .....   | 630 |
| 儲存點 .....   | 631 |
| 堆積大小 .....  | 631 |
| 緩衝區消脹 .....                                       | 631 |

|   |     |
|---|-----|
| 可修改的 Flink 組態屬性 .....                           | 631 |
| 重啟策略 .....                                      | 632 |
| 檢查點和狀態後端 .....                                  | 632 |
| 檢查點 .....                                       | 632 |
| RockSDB 原生度量 .....                              | 632 |
| 進階狀態後端選項 .....                                  | 634 |
| 完整 TaskManager 選項 .....                         | 634 |
| 記憶體組態 .....                                     | 634 |
| RPC/阿卡 .....                                    | 635 |
| 用戶端 .....                                       | 635 |
| 進階叢集選項 .....                                    | 635 |
| 檔案系統組 .....                                     | 635 |
| 進階容錯選項 .....                                    | 636 |
| 記憶體組態 .....                                     | 634 |
| 指標 .....  | 636 |
| 端點和用戶REST端的進階選項 .....                           | 636 |
| 進階SSL安全性選項 .....                                | 636 |
| 進階排程選項 .....                                    | 636 |
| Flink 網頁使用者介面的進階選項 .....                        | 636 |
| 檢視已設定的 Flink 內容 .....                           | 636 |
| 為 Apache Flink 設定受管服務，以存取 Amazon 中的資源 VPC ..... | 638 |
| Amazon VPC 概念 .....                             | 638 |
| VPC應用程式權 .....                                  | 639 |
| 添加訪問 Amazon 的許可政策 VPC .....                     | 639 |
| 為 Apache Flink 應用程VPC式的連線受管理服務建立網際網路和服務存取 ..... | 640 |
| 相關資訊 .....                                      | 641 |
| 使用適用於阿帕奇 Flink 的受管理服務 VPC API .....             | 641 |
| 建立應用程式 .....                                    | 641 |
| AddApplicationVpcConfiguration .....            | 642 |
| DeleteApplicationVpcConfiguration .....         | 643 |
| 更新應用 .....                                      | 643 |
| 範例：使用 VPC .....                                 | 644 |
| 疑難排解 Apache Flink 的受管理服務 .....                  | 645 |
| 開發疑難排 .....                                     | 645 |
| 系統復原最佳做法 .....                                  | 645 |
| 胡迪配置最佳實踐 .....                                  | 646 |

|   |     |
|---|-----|
| Apache Flink 火焰圖 .....                            | 647 |
| EFO連接器 1.15.2 的認證提供者問題 .....                      | 647 |
| 應用程式使用不支援的 Kinesis 連接器 .....                      | 647 |
| 編譯錯誤：「無法解析項目的依賴關係」 .....                          | 650 |
| 無效的選擇：「動力分析 Sv2」 .....                            | 650 |
| UpdateApplication 動作不重新加載應用程序代碼 .....             | 650 |
| S3 StreamingFileSink FileNotFoundExceptions ..... | 651 |
| FlinkKafkaConsumer 與保存點停止問題 .....                 | 652 |
| FLINK 1.15 非同步接收器死鎖 .....                         | 652 |
| 在重新分片期間，Amazon Kinesis 資料串流來源處理不順序 .....          | 662 |
| 運行時疑難 .....                                       | 662 |
| 故障診斷工具 .....                                      | 663 |
| 應用問題 .....  | 663 |
| 應用程式重新啟 .....                                     | 667 |
| 輸送量太慢 .....                                       | 669 |
| 无界状态增长 .....                                      | 670 |
| I/O 綁定運算子 .....                                   | 671 |
| Kinesis 資料串流的上游或來源限流 .....                        | 671 |
| 檢查點 .....   | 672 |
| 檢查點逾時 .....                                       | 678 |
| 檢查點失敗的阿帕奇梁 .....                                  | 679 |
| 背壓 .....  | 681 |
| 資料扭曲 .....  | 682 |
| 狀態扭曲 .....  | 682 |
| 與不同地區的資源整合 .....                                  | 683 |
| 文件歷史紀錄 .....                                      | 684 |
| API 範例程式碼 .....                                   | 689 |
| AddApplicationCloudWatchLoggingOption .....       | 690 |
| AddApplicationInput .....                         | 690 |
| AddApplicationInputProcessingConfiguration .....  | 691 |
| AddApplicationOutput .....                        | 692 |
| AddApplicationReferenceDataSource .....           | 692 |
| AddApplicationVpcConfiguration .....              | 693 |
| CreateApplication .....                           | 693 |
| CreateApplicationSnapshot .....                   | 695 |
| DeleteApplication .....                           | 695 |

|   |     |
|---|-----|
| DeleteApplicationCloudWatchLoggingOption .....      | 695 |
| DeleteApplicationInputProcessingConfiguration ..... | 695 |
| DeleteApplicationOutput .....                       | 696 |
| DeleteApplicationReferenceDataSource .....          | 696 |
| DeleteApplicationSnapshot .....                     | 696 |
| DeleteApplicationVpcConfiguration .....             | 696 |
| DescribeApplication .....                           | 697 |
| DescribeApplicationSnapshot .....                   | 697 |
| DiscoverInputSchema .....                           | 697 |
| ListApplications .....                              | 698 |
| ListApplicationSnapshots .....                      | 698 |
| StartApplication .....                              | 698 |
| StopApplication .....                               | 699 |
| UpdateApplication .....                             | 699 |
| API 參考 .....  | 700 |
| .....   | 701 |

Amazon Managed Service for Apache Flink 之前稱為 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。



## 什麼是阿帕奇 Flink 的 Amazon 託管服務？

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

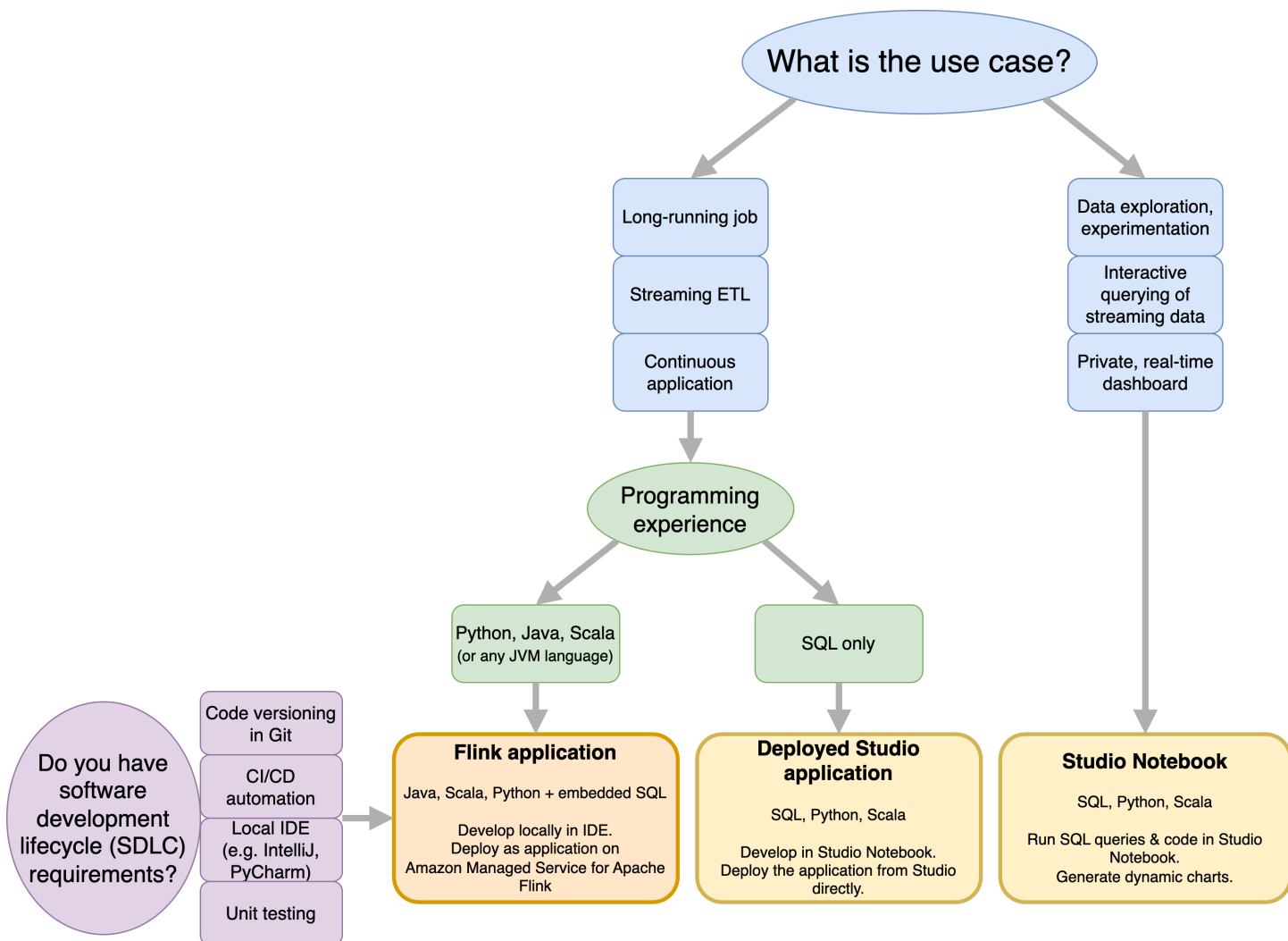
[您可以使用以 Apache Flink 為基礎的開放原始碼程式庫，在 Apache Flink 的受管理服務中，使用您選擇的語言來建置應用程式。](#) Apache Flink 是處理資料串流的熱門框架及引擎。

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。

## 決定使用阿帕奇 Flink 的管理服務或管理服務為阿帕奇 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。使用 [Apache Flink 的託管服務](#)，您可以使用您選擇的和 Apache Flink 數據流或表格以 Java，斯卡拉或 Python (和嵌入式SQL) 構建 Flink 應用程式。IDE APIs 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式，該應用程式將承擔諸如流ETL或連續應用程式之類的工作負載，則應考慮使用 [Apache Flink 的託管服務](#)。這是因為您可以APIs直接在您選擇的情況下使用 Flink 建立您IDE的 Flink 應用程式。使用您的在本機開發IDE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 中的程式碼版本控制、CI/CD 自動化或單元測試。

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。熟悉的使用者SQL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，如果您想要整合您的SDLC工具，例如 Git 和 CI/CD 自動化上的程式碼版本控制，或是單元測試等技術，我們建議您使用您選擇的 Apache Flink 管理服務。IDE

## 選擇要在 Apache Flink APIs 的受管理服務中使用的 Apache Flink

您可以使用 Java，Python 和斯卡拉在 Apache Flink 的託管服務中使用 Apache Flink APIs 在您選擇 IDE 的一個應用程序。[您可以API在文件中找到有關如何使用 Flink 資料流和表格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，APIs 以及最符合應用程式和作業需求的語言。如果您不確定，本節將提供高階指引以協助您。

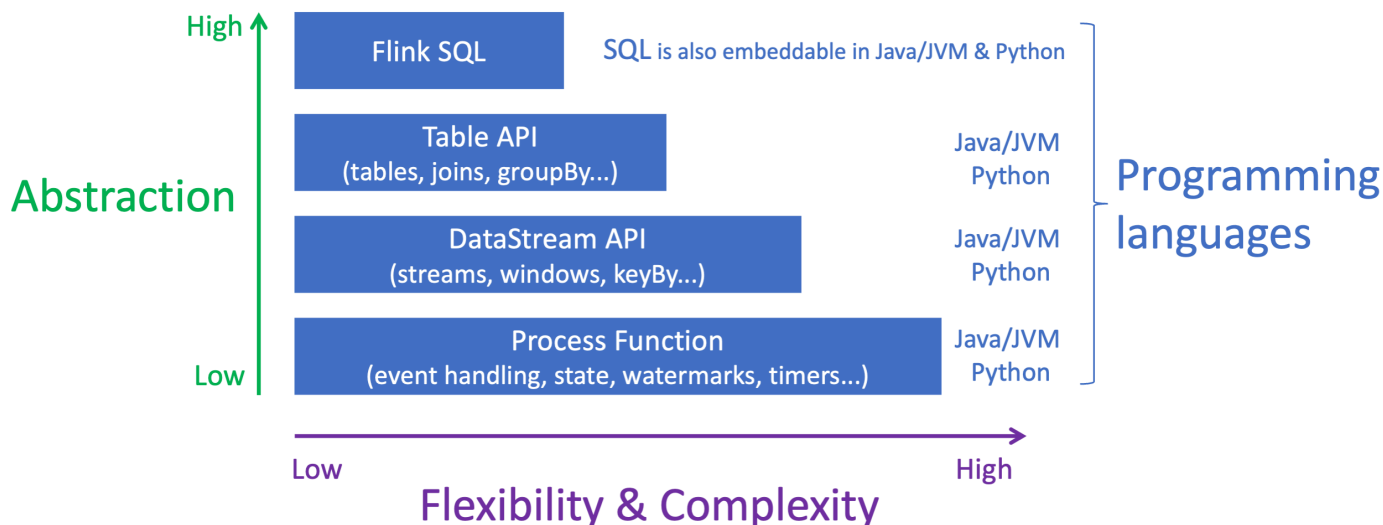
### 選擇一個快速連結 API

Apache Flink APIs 具有不同級別的抽象，可能會影響您決定構建應用程序的方式。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。您不必只使用一個 Flink API。您可以APIs在 [Apache Flink 文件中進一步了解有關 Flink](#) 的資訊。

Flink 提供了四個級別的API抽象：Flink SQLAPI，表 DataStream API，和過程函數，這是與 DataStream API 這些都在阿帕奇 Flink 的 Amazon 託管服務中受到支持。建議盡可能從更高層級的抽象開始，但是某些 Flink 功能僅適用於[數據流](#)，您可以在API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）
- 您想要使用自定義計時器（例如實現自定義窗口或後期事件處理）
- 您希望能夠在不重置狀態的情況下修改應用程序的流程

## Apache Flink APIs



### Note

選擇具有以下內容的語言 DataStreamAPI：

- SQL無論選擇的程式語言為何，都可以嵌入任何 Flink 應用程式中。
- 如果您打算使用 DataStream API，則 Python 中並非所有連接器都受到支援。
- 如果你需要低延遲/高吞吐量，你應該考慮使用 Java/Scala 而不管 API
- 如果您打算在過程函數中使用異步 IO，API則需要使用 Java。

的選擇也會影響您在API不必重設狀態的情況下進化應用程式邏輯的能力。這取決於特定功能(在運算子UID上設定的能力)，該功能僅DataStreamAPI適用於 Java 和 Python。如需詳細資訊，請參閱 Apache Flink 文件中的[UIDs針對所有運算子設定](#)。

## 開始使用串流資料應用程式

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IDE的程式碼撰寫程式碼，並使用即時串流資料進行測試。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：

- [阿帕奇 Flink 的託管服務：它是如何工作的](#)
- [開始使用阿帕奇 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。只要在中按幾下滑鼠 AWS Management Console，即可啟動無伺服器筆記本，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：

- [使用具有 Apache Flink 管理服務的工作室筆記本](#)
- [建立工作室筆記本](#)

# 阿帕奇 Flink 的託管服務：它是如何工作的

適用於 Apache Flink 的受管服務是一項全受管的 Amazon 服務，可讓您使用 Apache Flink 應用程式來處理串流資料。首先，您編程您的 Apache Flink 應用程序，然後創建 Apache Flink 應用程序的託管服務。

## 編程您的阿帕奇 Flink 應用程序

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 [表格 API](#)。另一個 Apache Flink APIs 也可供您使用，但它們在構建流應用程序時不常用。

兩者的特點 APIs 如下：

### DataStream API

阿帕奇 Flink DataStream API 編程模型是基於兩個組成部分：

- 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。

使用建立的應用程式 DataStream API 請執行下列動作：

- 從資料來源讀取資料 (例如 Kinesis 串流或 Amazon MSK 主題)。
- 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。

使用的應用程式 DataStream API 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。

若要取得有關使用的更多資訊 DataStream API，請參閱[檢閱 DataStream API 元件](#)。

## 表 API

Apache 的 Flink 表格API程式設計模型是基於下列元件：

- 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。
- 表格：提供資料SQL表或檢視表存取權的物件。
- 資料表來源：用於讀取外部來源的資料，例如 Amazon MSK 主題。
- 資料表函數：用來轉換資料的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。

使用表格建立的應用程式會API執行下列動作：

- 透過連線至 Table Source 建立 TableEnvironment。
- 在TableEnvironment使用SQL查詢或表API函數中建立表格。
- 使用資料表API或在資料表上執行查詢 SQL
- 使用表函數或查詢在查詢結果上應用SQL轉換。
- 將查詢或函數結果寫入 Table Sink。

使用該表的應用程式API可以用 Java 或 Scala 編寫，並且可以使用API調用或SQL查詢查詢數據。

若要取得有關使用表格的更多資訊API，請參閱[檢閱表格API元件](#)。

## 為 Apache Flink 應用程式建立您的受管理服務

適用於 Apache Flink 的受管理服 AWS 務是一項服務，可建立託管 Apache Flink 應用程式的環境，並提供下列設定：

- [在 Apache Flink 的受管理服務中使用執行階段屬性](#)：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。
- [在 Apache Flink 的受管理服務中實作容錯能力](#)：應用程式如何從中斷和重新啟動中復原。

- 在 [Amazon 管理服務中為 Apache Flink 進行日誌記錄和監控](#)：您的應用程式如何將事件記 CloudWatch 錄到記錄檔。
- 在 [Apache Flink 的受管理服務中實作應用程式調整](#)：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 [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#)。

## 為 Apache Flink 應用程式建立受管理服務

本主題包含有關為 Apache Flink 應用程式建立受管理服務的資訊。

本主題包含下列章節：

- [建立適用於 Apache Flink 應用程式程式碼的託管服務](#)
- [為 Apache Flink 應用程式建立您的受管理服務](#)
- [啟動適用於 Apache Flink 應用程式的管理服務](#)
- [驗證您的 Apache Flink 應用程式託管服務](#)
- [為您的 Apache Flink 應用程式的受管理服務啟用系統復原](#)

## 建立適用於 Apache Flink 應用程式程式碼的託管服務

本節說明您用來為 Apache Flink 應用程式的受管理服務建置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。如需升級 Managed Service for Apache Flink 應用程式的相關資訊，請參閱 [為 Apache Flink 使用就地版本升級](#)。

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

適用於 Apache Flink 的受管理服務支援最大 512 MB 的 JAR 檔案。如果您使用的 JAR 檔案大於此，您的應用程式將無法啟動。

應用程式現在可以使用任何 Scala 版本的 Java。您必須將您選擇的 Scala 標準庫捆綁到您的 Scala 應用程式中。



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 [為 Apache Flink 應用程式使用具有託管服務的 Apache 光束](#)。

## 指定應用程式的 Apache Flink 版本

使用 Managed Service for Apache Flink 執行期 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。例如，如果您使用的是阿帕奇 Flink 1.19.1，請提供以下內容：

```
mvn package -Dflink.version=1.19.1
```

若要使用舊版 Apache Flink 建置應用程式，請參閱 [〈〉 早期版本](#)。

## 為 Apache Flink 應用程式建立您的受管理服務

建立應用程式程式碼之後，請執行下列動作來建立 Apache Flink 應用程式的受管理服務：

- 上傳應用程式的程式碼：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 [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#) 教學課程。
- 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：
  - 使用主控台建立 Apache Flink 應用程式的受管理服務：您可以使用 AWS 主控台建立和設定應用程式 AWS 式。

當您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 記錄資料流、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 [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#) 教學課程。

- 使用以下指令建立 Apache Flink 應用程式的受管理服務 AWS CLI：您可以使用 AWS CLI。

使用建立應用程式時 CLI，您也必須手動建立應用程式的相依資源 (例如 CloudWatch 記錄資料流、IAM 角色和 IAM 原則)。

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

#### Note

您可以變更現RuntimeEnvironment有應用程式的。如要瞭解如何作業，請參閱[為 Apache Flink 使用就地版本升級](#)。

## 啟動適用於 Apache Flink 應用程式的管理服務

建置應用程式的程式碼、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：

- 使用主控台啟動 Apache Flink 應用程式的受管理服務：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 AWS 式。
- 使用以下命令啟動 Apache Flink 應用程式的受管理服務 AWS API：您可以使用[StartApplication](#)動作執行應用程式。

## 驗證您的 Apache Flink 應用程式託管服務

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[在 Amazon 管理服務中為 Apache Flink 進行日誌記錄和監控](#)。
- 使用 CloudWatch 指標：您可以使用 CloudWatch 指標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、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。

## 為您的 Apache Flink 應用程式的受管理服務啟用系統復原

透過系統復原功能，您可以在適用於 Apache Flink 的 Amazon 受管服務上，達到執行中的 Apache Flink 應用程式的更高可用性。選擇使用此組態可讓服務在執行UpdateApplication或執行程式碼或組態錯誤等動作時，自動將應用程式還原為先前autoscaling執行的版本。

### Note

若要使用系統還原功能，您必須透過更新應用程式來選擇加入。現有的應用程式預設不會自動使用系統復原。

### 運作方式

當您啟動應用程式作業 (例如更新或擴展動作) 時，Apache Flink 的 Amazon 受管服務會先嘗試執行該作業。如果偵測到造成作業無法成功發生的問題 (例如程式碼錯誤或權限不足)，服務就會自動啟動作RollbackApplication業。

復原會嘗試將應用程式還原到成功執行的先前版本，以及相關的應用程式狀態。如果復原成功，您的應用程式會使用舊版繼續以最短的停機時間來處理資料。如果自動復原也失敗，適用於 Apache Flink 的 Amazon 受管服務會將應用程式轉換為READY狀態，以便您可以採取進一步的動作，包括修正錯誤和重試作業。

您必須選擇加入才能使用自動系統復原。您可以使用控制台或從這一點開始API對應用程序上的所有操作啟用它。

下列UpdateApplication動作要求範例會啟用應用程式的系統復原：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationSystemRollbackConfigurationUpdate": {
      "RollbackEnabledUpdate": "true"
    }
  }
}
```

### 檢閱自動系統復原的常見案例

下列案例說明自動系統復原的好處：

- 應用程式更新：如果您在透過 main 方法初始化 Flink 工作時使用新程式碼更新應用程式有錯誤，則自動復原會允許還原先前的工作版本。其他有助於系統復原的更新案例包括：
  - 如果您的應用程式已更新為以高於 [maxParallelism](#)。
  - 如果您的應用程式已更新為在 Flink 工作啟動期間導致失敗的VPC應用程式使用不正確的子網路執行。
- Flink 版本升級：當您升級至新的 Apache Flink 版本且升級的應用程式遇到快照相容性問題時，系統還原可讓您自動還原至先前的 Flink 版本。
- AutoScaling：由於快照與 Flink 工作圖表之間的運算子不符，應用程式擴展但遇到從儲存點還原的問題。

## 使用作業APIs進行系統復原

為了提供更好的可見性，適用於 Apache Flink 的 Amazon 受管服務有兩個APIs與應用程式操作有關，可協助您追蹤故障和相關系統復原。

### ListApplicationOperations

這會以反向時間順序API列出在應用程式上執行的所有作業 UpdateApplication MaintenanceRollbackApplication，包括、和其他作業。下

列ListApplicationOperations動作要求範例會列出應用程式的前 10 個應用程式作業：

```
{
  "ApplicationName": "MyApplication",
  "Limit": 10
}
```

下列範例要求ListApplicationOperations可協助將清單篩選為應用程式上先前的更新：

```
{
  "ApplicationName": "MyApplication",
  "operation": "UpdateApplication"
}
```

### DescribeApplicationOperation

這會API提供有關列出之特定作業的詳細資訊ListApplicationOperations，包括失敗原因 (如果適用)。下列DescribeApplicationOperation動作要求範例會列出特定應用程式作業的詳細資訊：

```
{
  "ApplicationName": "MyApplication",
  "OperationId": "xyzoperation"
}
```

如需故障診斷資訊，請參閱[系統復原最佳做法](#)。

## 為阿帕奇 Flink 應用程式運行託管服務

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 for 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。

如需有關 Apache Flink 工作的詳細資訊，請參閱 Apache Flink 文件中的[工作和排程](#)。

## 識別申請和工作狀態

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，描述其執行期。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。如果應用程式保持暫時狀態一段時間，您可以使用Force參數設定為的[StopApplication](#)動作來停止應用程式true。這些狀態包括下列項目：
    - STARTING：[StartApplication](#)動作之後發生。應用程式正在從狀態 READY 轉換為 RUNNING 狀態。
    - STOPPING：[StopApplication](#)動作之後發生。應用程式正在從狀態 RUNNING 轉換為 READY 狀態。
    - DELETING：[DeleteApplication](#)動作之後發生。正在刪除應用程式。
    - UPDATING：[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 `AutoScalingEnabled` 屬性 [ParallelismConfiguration](#) 集為 `true` , 而且服務正在增加應用程式的平行處理原則。當應用程式處於此狀態時, 您唯一可以使用的有效 [API StopApplication](#) 動作是 `Force` 參數設定為 `true` 的動作。如需自動擴展的相關資訊, 請參閱 [在適用於 Apache Flink 的受管理服務中使用自動調整](#)。
- **FORCE\_STOPPING**: 在將 `Force` 參數設定為 `true` 的情況下呼叫 [StopApplication](#) 動作之後發生 `true`。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**: [RollbackApplication](#) 動作被調用後發生。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。
- **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 [管理適用於 Apache Flink 的受管理服務的維護工作](#)。

您可以使用主控台或使用 [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期。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狀態變更 CloudWatch 記錄, 以檢查工作的狀態。

## 執行批次工作量

Managed Service for Apache Flink 支援執行 Apache Flink 批次工作負載。在批次工作中, 當 Apache Flink 工作進入 `FINISHED` 狀態時, Apache Flink 應用程式的受管理服務狀態會設定為 `READY`。如需 Flink 作業狀態的詳細資訊, 請參閱 [作業與排程](#)。

## 檢閱 Apache Flink 應用程式資源的受管理服務

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。

## 阿帕奇 Flink 應用程式資源的管理服務

適用於 Apache Flink 的受管理服務是一項 AWS 服務，可為您的 Apache Flink 應用程式建立一個環境。適用於 Apache Flink 服務的受管理服務會使用稱為 Kinesis 處理單元 (KPU) 的單位來提供資源。

其中一個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

KPUs在不同的執行單元中執行應用程式，稱為工作和子工作。你可以將一個子任務視為等效於一個執行緒。

應用程式KPUs可用的數目等於應用程式的Parallelism設定，除以應用程式的ParallelismPerKPU設定。

如需應用程式平行處理的詳細資訊，請參閱 [在 Apache Flink 的受管理服務中實作應用程式調整](#)。

## 阿帕奇 Flink 應用程序資源

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。當 Apache Flink 的受管理服務為您的應用程式配置資源時，它會將一個或多個 Apache Flink 任務插槽指派給單一位置。KPU分配給單個插槽的數量等於KPU於您的應用程序的ParallelismPerKPU設置。如需有關 [Job 位置的詳細資訊](#)，請參閱 [Apache Flink 文件中的作業排程](#)。

## 操作員並行

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。

如需有關運算子的詳細資訊，請參閱 Apache Flink 文件中的[運算子](#)。

## 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，則執行期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 主控台內的作業圖表。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。

## 檢閱 DataStream API 元件

您的 Apache Flink 應用程式會使用 [Apache Flink DataStream API](#) 來轉換資料串流中的資料。

本節說明移動、轉換和追蹤資料的不同元件：

- [使用連接器在適用於 Apache Flink 的受管理服務中移動資料 DataStream API](#)：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。
- [使用 Apache Flink 受管理服務中的運算子來轉換資料 DataStream API](#)：這些元件可以轉換或分組應用程式中的資料元素。
- [在 Apache Flink 的受管理服務中使用 DataStream API](#)：本主題說明 Apache Flink 的受管理服務如何在 DataStream API 使用。

## 使用連接器在適用於 Apache Flink 的受管理服務中移動資料 DataStream API

在適用於 Apache Flink 的 Amazon 受管服務中 DataStream API，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：



- [將串流資料來源新增至 Apache Flink 的受管理服務](#)：從 Kinesis 資料串流、檔案或其他資料來源向應用程式提供資料。
- [在 Apache Flink 的受管理服務中使用接收器寫入資料](#)：將資料從您的應用程式傳送到 Kinesis 資料串流、Firehose 串流或其他資料目的地。
- [在 Apache Flink 的受管理服務中使用非同步 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](#) 中的 [連接器](#)。

### Warning

如果您的應用程式在 Flink 1.6、1.8、1.11 或 1.13 上執行，且想要在中東 (UAE)、亞太區域 (海德拉巴)、以色列 (特拉維夫)、歐洲 (蘇黎世)、中東 ()、亞太區域 (墨爾本 UAE) 或亞太區域 (雅加達) 區域執行，您可能需要使用更新的連接器重建應用程式封存檔，或升級至 Flink 1.18。

Apache Flink 連接器會儲存在它們自己的開放原始碼儲存庫中。如果您要升級到版本 1.18 或更高版本，則必須更新依賴項。若要存取 Apache Flink AWS 連接器的儲存庫，請參閱 [flink-connector-aws](#)。

以下是建議的指導方針：

### 連接器升級

| 使用的連接器 | 解析度  |
|--------|--|
| 1 EFO  | 升級至 Apache Flink 1.15 版的 Amazon 受管服務時，請確定您使用的是最新 EFO 的連接器。這必須是任何版本 1.15.3 或更高版本。如需詳細資訊，請參閱：<br><br><a href="#">FLINK-29324</a> 。 |

| Flink 使用的連接器            | 解析度  |
|-------------------------|--|
| 1 Amazon 數據 Firehose 收器 | <p>升級為 Apache Flink 1.15 版的 Amazon 託管服務時，請確保您使用的是最新的 Amazon 數據 Firehose 接收器。</p> <p><a href="#">Amazon 數據 Firehose 收器</a></p>  |
| 1 卡夫卡連接器                | <p>升級為 Apache Flink 版本 1.15 的 Amazon 託管服務時，請確保您使用的是最新的卡夫卡連接器。APIs 阿帕奇 Flink 已棄用 <a href="#">FlinkKafkaConsumer</a> 和 <a href="#">FlinkKafkaProducer</a>。這些 APIs 用於卡夫卡下沉的卡夫卡不能承諾 Flink 1.15。請確定您正在使用 <a href="#">KafkaSource</a> 和 <a href="#">KafkaSink</a>。</p> |
| 1 Firehose              | <p>您的應用程式取決於舊版 Firehose 連接器，而該版本不知道較新的 AWS 區域。使用 Firehose 連接器 2.1.0 版重建應用程式存檔。</p> <p><a href="#">2.1.0 版</a></p>  |

## F 使用的連接器

## 解析度

## 1 Kinesis

您的應用程式取決於過時版本的 Flink Kinesis 連接器，而該連接器不知道較新 AWS 的區域。使用 Flink Kinesis 連接器 1.6.1 版重建應用程式存檔。

<https://github.com/aws-labs/amazon-kinesis-connector-flink/tree/1.6.1>

## 1 Kinesis

您的應用程式取決於過時版本的 Flink Kinesis 連接器，而該連接器不知道較新 AWS 的區域。使用 Flink Kinesis 連接器 2.4.1 版重建應用程式存檔。

<https://github.com/aws-labs/amazon-kinesis-connector-flink/tree/2.4.1>

## 1 Kinesis

您的應用程式取決於過時版本的 Flink Kinesis 連接器，而該連接器不知道較新 AWS 的區域。很抱歉，Flink 不再發行 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，以更新至 Flink 1.15。

## 將串流資料來源新增至 Apache Flink 的受管理服務

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

使用 Kinesis 資料串流

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));
```

如需如何使用 `FlinkKinesisConsumer` 的詳細資訊，請參閱 [下載並檢查阿帕奇 Flink 流 Java 代碼](#)。

### 創建使 `FlinkKinesisConsumer` 用 EFO 消費者

`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，而不是讓消費者與其他從串流讀取的消費者共用串流的固定頻寬。

如需與 Kinesis 取用者 EFO 搭配使用的詳細資訊，請參閱 [FLIP-128：適用於 AWS Kinesis 消費者的增強型扇出](#)。

您可以在 Kinesis 取用者 EFO 者上設定下列參數來啟用取用者：

- `RECORD_PUBLISHER_TYPE`：將此參數設定為 EFO，讓您的應用程式使用取用者存取 Kinesis 資料串流資料。
- `EFO_CONSUMER_NAME`：將此參數設置為此流的消費者中唯一的字符串值。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` 要使用的 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");
```

如需使用取用EFO者之 Apache Flink 應用程式的受管理服務範例，請參閱[增強型扇出 \(EFO\) 取用者](#)。

使用 Amazon MSK

來KafkaSource源會從 Amazon MSK 主題向您的應用程式提供串流資料。

## 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");
```

如需如何使用 KafkaSource 的詳細資訊，請參閱 [MSK 複寫](#)。

## 在 Apache Flink 的受管理服務中使用接收器寫入資料

在應用程式程式碼中，您可以使用任何 [Apache Flink 接收器](#) 連接器寫入外部系統，包括 AWS 服務，例如 Kinesis Data Streams 和 DynamoDB。

Apache Flink 還為文件和套接字提供接收器，您可以實現自定義接收器。在幾個受支持的接收器中，以下是經常使用的：

使用 Kinesis 資料串流

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 [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#)。

## 使用阿帕奇卡夫卡和 Amazon Managed Streaming for Apache Kafka () MSK

[Apache Flink 卡夫卡連接器](#)為將數據發布到 [Apache 卡夫卡](#)和 Amazon MSK 提供了廣泛的支持，包括僅一次保證。要了解如何寫信給卡夫卡，請參閱 Apache Flink 文檔中的 [卡夫卡連接器示例](#)。

## 使用 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”](#)。

## 使用 Firehose

`FlinkKinesisFirehoseProducer`這是一個可靠、可擴充的 Apache Flink 接收器，用於使用 [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及設定 Apache Flink 資料串流中的資料，`FlinkKinesisFirehoseProducer`並將資料傳送至 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");
}
```

如需有關如何使用 Firehose 接收槽的完整教學課程，請參閱[the section called “Firehose 水槽”](#)。

## 在 Apache Flink 的受管理服務中使用非同步 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[非同步 I/O](#)。

## 使用 Apache Flink 受管理服務中的運算子來轉換資料 DataStream API

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 文件中的[運算子](#)。

本主題包含下列章節：

- [使用轉換運算子](#)
- [使用彙總運算子](#)

### 使用轉換運算子

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");  
        }  
    }  
);
```

## 使用彙總運算子

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;  
    }));
```

如需更多程式碼範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

## 在 Apache Flink 的受管理服務中使用 DataStream API

Managed Service for 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。

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

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

如需有關時間戳記的詳細資訊，請參閱 Apache Flink 文件中的「[產生浮水印](#)」。

## 檢閱表格API元件

您的 Apache Flink 應用程式會使用 [Apache Flink 表格 API](#)，透過關聯式模型與串流中的資料互動。您可以使用表格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：

- [使用表格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。
- [檢閱表格API時間屬性](#)：本主題說明 Apache Flink 的受管理服務如何在使用表格API時追蹤事件。

## 使用表格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 表格API，您可以使用下列類型的連接器：

- [表格API來源](#)：您可以使用表格API來源連接器，在您TableEnvironment使用API呼叫或SQL查詢建立資料表。
- [表API水槽](#)：您可以使用SQL命令將表格資料寫入外部來源，例如 Amazon MSK 主題或 Amazon S3 儲存貯體。

## 表格API來源

您可以從資料串流建立資料表來源。下面的代碼從 Amazon MSK 主題創建一個表：

```
//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);
```

如需有關資料表來源的詳細資訊，請參閱 Apache Flink 文件中的資料[表與SQL連接器](#)。

## 表API水槽

若要將資料表資料寫入接收器，請在中建立接收器SQL，然後在StreamTableEnvironment物件上執行SQL基於接收器的接收器。

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 文件中的[支援連接器](#)。

## 使用者定義來源和匯

您可以使用現有的 Apache Kafka 連接器，將資料傳送至其他 AWS 服務，例如 Amazon MSK 和 Amazon S3。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 文件中的[使用者定義來源和接收器](#)。

## 檢閱表格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 表格根據記錄時間API建立視窗時，您可以使用方法來定義其使用哪些時間戳記 `setStreamTimeCharacteristic`。

如需有關將時間戳記與資料表搭配使用的詳細資訊API，請參閱 Apache Flink 文件中的[時間屬性](#)和[即時資料流處理](#)。

## 使用 Python 與阿帕奇 Flink 的託管服務

### 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。如需相關[step-by-step 指示](#)，請參閱[蘋果矽 Mac 的 PyFlink 1.15 開發](#)。

阿帕奇 Flink 版本 1.18.1 包括使用 Python 版本 3.10 創建應用程式的支持。如需詳細資訊，請參閱[Flink Python 文件](#)。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。

在高層次上，Python 表API是圍繞 Java 表的包裝API。如需有關 Python 資料表的詳細資訊API，請參閱 Apache Flink 文件中的資料[表API教學課程](#)。

## 為阿帕奇 Flink Python 應用程式編程您的託管服務

您可以使用阿帕奇 Flink Python 應用程式編碼管理服務為 Apache Flink 的 Python 應用程式。API阿帕奇 Flink 引擎將 Python 表API語句（在 Python 虛擬機中運行）轉換為 Java 表API語句（在 Java 虛擬機中運行）。

您可以通過執行以下操作API來使用 Python 表：

- 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 Apache Flink 的受管理服務API中使用 Python 資料表，請參閱[開始使用適用於 Python 的阿帕奇 Flink 的 Amazon 託管服務](#)。

## 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建立SQL查詢的使用者定義函數。查SQL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前面的CreateTableSQL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 a StatementSet 與表互動，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))
```

## 讀取運行時屬性

您可以使用執行期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期屬性：

- 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 執行期所建立的名為 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KEY從上一個範例傳回的屬性群組讀取名為 INPUT STREAM \_\_ 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文件中的用戶定義 Java 代碼
- 檔案中的 Java 程JAR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。

## 為阿帕奇 Flink Python 應用程序創建您的託管服務

### 指定您的程式碼檔

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，您可以使用控制台或[CreateApplication](#)操作來創建應用程序。

當您使用[CreateApplication](#)動作建立應用程式時，您可以使用名為的特殊應用程式內容群組，在 zip 檔案中指定程式碼檔案和封存檔kinesis.analytics.flink.run.options。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。
- JAR 文件：包含 Java 用戶定義函數的 Java JAR 文件。
- pyFiles：一個 Python 資源文件，其中包含應用程序使用的資源。



- `pyArchives`: 包含應用程式資源檔案的 zip 檔案。

如需有關 Apache Flink Python 程式碼檔案類型的詳細資訊，請參閱 Apache Flink 文件中的[命令列介面](#)。

#### 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"
        }
      }
    ],
  },
},
```

## 監視您的阿帕奇 Flink Python 應用程式的託管服務

您可以使用應用程式的 CloudWatch 記錄檔來監視 Apache Flink Python 應用程式的受管理服務。

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。

## 查詢日誌與 CloudWatch 見解

下列 CloudWatch 見解查詢會在執行應用程式的主要功能時，搜尋 Python 入口點所建立的記錄檔：

```
fields @timestamp, message
| sort @timestamp asc
```

```
| filter logger like /PythonDriver/  
| limit 1000
```

## 在 Apache Flink 的受管理服務中使用執行階段屬性

您可以使用執行期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：

- [使用主控台管理執行階段屬性](#)
- [使用管理執行階段屬性 CLI](#)
- [在 Apache Flink 應用程式的受管理服務中存取執行階段屬性](#)

### 使用主控台管理執行階段屬性

您可以使用在 AWS Management Console Apache Flink 應用程式中新增、更新或移除執行階段屬性。

#### Note

如果您使用的是較早支援的 Apache Flink 版本，並且想要將現有的應用程式升級到 Apache Flink 1.19.1，您可以使用就地 Apache Flink 版本升級來執行此操作。透過就地版本升級，您可以以 ARN 跨 Apache Flink 版本保留應用程式可追溯性，包括快照、記錄、指標、標籤、Flink 組態等。您可以在 RUNNING 和 READY 狀態中使用此功能。如需詳細資訊，請參閱 [為 Apache Flink 使用就地版本升級](#)。

### 更新 Managed Service for Apache Flink 應用程式的執行期屬性

1. 在 <https://console.aws.amazon.com/flink> 開啟適用於阿帕奇 Flink 的受管理服務
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，選擇設定。
4. 展開屬性區段。
5. 使用屬性區段中的控制項，以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期屬性。
6. 選擇更新。

## 使用管理執行階段屬性 CLI

您可以使用 [AWS CLI](#) 新增、更新或移除執行期屬性。

本節包含設定應用程式執行階段屬性的API動作要求範例。若要取得有關如何使用JSON檔案進API行動作輸入的資訊，請參閱 [〈〉 適用於阿帕奇 Flink API 範例程式碼的託管服務](#)。

### Note

取代範例帳戶 ID (*012345678901*) 在下列範例中顯示您的帳戶 ID。

### 建立應用程式時新增執行階段屬性

[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期屬性群組 (ProducerConfigProperties 和 ConsumerConfigProperties)：

```
{
  "ApplicationName": "MyApplication",
  "ApplicationDescription": "my java test app",
  "RuntimeEnvironment": "FLINK-1_19",
  "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階段屬性

[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期屬性：

```
{
  "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期屬性，Managed Service for Apache Flink 會將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期屬性，Managed Service for Apache Flink 會更新屬性值。

## 刪除運行時屬性

[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期屬性和屬性群組：

```
{
  "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。

## 在 Apache Flink 應用程式的受管理服務中存取執行階段屬性

您可以使用可傳回 `Map<String, Properties>` 物件的靜態

`KinesisAnalyticsRuntime.getApplicationProperties()` 方法，在 Java 應用程式的程式碼中擷取執行期屬性。

下列 Java 程式碼範例會擷取應用程式的執行期屬性：

```
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期屬性擷取的 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;
}
```

如需程式碼範例，請參閱「[為 Apache Flink 應用程式建立及使用受管理服務的範例](#)」。

## 搭配 Apache Flink 的受管理服務使用 Apache Flink 連接器

Apache Flink 連接器是一種軟體元件，可將資料移入和移出適用於 Apache Flink 應用程式的 Amazon 受管服務。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、Apache Kafka 主題、檔案或其他資料來源將資料提供給您的應用程式。
- 接收器：將資料從您的應用程式傳送到 Kinesis 資料串流、Firehose 串流、Apache Kafka 主題或其他資料目的地。
- 非同步 I/O：提供對資料來源 (例如資料庫) 的非同步存取，以豐富串流。

Apache Flink 連接器會儲存在它們自己的來源儲存庫中。Apache Flink 連接器的版本和成品會根據您使用的 Apache Flink 版本以及您使用的是 DataStream、表格還是而有所不同。SQL API

適用於 Apache Flink 的 Amazon 受管服務支援超過 40 個預先建置的 Apache Flink 來源和接收器連接器。下表提供最常用的連接器及其相關版本的摘要。您也可以使用非同步接收器架構建立自訂接收器。如需詳細資訊，請參閱 Apache Flink 文件中的 [一般非同步基本接收器](#)。

若要存取 Apache Flink AWS 連接器的儲存庫，請參閱[flink-connector-aws](#)。

### 適用於 Flink 版本的連接器

| 連接器                               | 菲林克 1.15 版                               | 菲林克 1.18 版                               | 菲林克 1.19 版                               |
|-----------------------------------|--|--|--|
| Kinesis 資料流-來源-DataStream 和表 API  | flink-connector-kinesis,                 | flink-connector-kinesis, 4.3.0-1.18      | flink-connector-kinesis, 4.3.0-1.19      |
| Kinesis 數據流-接收器-DataStream 和表 API | flink-connector-aws-kinesis-串流           | flink-connector-aws-kinesis-數據流，         | flink-connector-aws-kinesis-資料流，         |
| Kinesis Data Streams-來源/接收-SQL    | flink-sql-connector-kinesis,             | flink-sql-connector-kinesis, 4.3.0-1.18  | flink-sql-connector-kinesis, 4.3.0-1.19  |
| 卡夫卡-DataStream 和表 API             | flink-connector-kafka,                   | flink-connector-kafka, 3.2.0-1.18        | flink-connector-kafka, 3.2.0-1.19        |
| 卡夫卡-SQL                           | flink-sql-connector-kafka,               | flink-sql-connector-kafka, 3.2.0-1.18    | flink-sql-connector-kafka, 3.2.0-1.19    |
| Firehose- DataStream 和桌子 API      | flink-connector-aws-kinesis-防火喉          | flink-connector-aws-firehose, 4.3.0-1.18 | flink-connector-aws-firehose, 4.3.0-1.19 |
| Firehose-SQL                      | flink-sql-connector-aws-運動-防火喉, 1.15.4   | flink-sql-connector-aws-消防喉管             | flink-sql-connector-aws-防火喉,             |
| DynamoDB- DataStream 和表 API       | flink-connector-dynamodb, 3.0.0-1.15     | flink-connector-dynamodb, 4.3.0-1.18     | flink-connector-dynamodb, 4.3.0-1.19     |
| DynamoDB-SQL                      | flink-sql-connector-dynamodb, 3.0.0-1.15 | flink-sql-connector-dynamodb, 4.3.0-1.18 | flink-sql-connector-dynamodb, 4.3.0-1.19 |



| 連接器                               | 菲林克 1.15 版 | 菲林克 1.18 版   | 菲林克 1.19 版   |
|-----------------------------------|------------|--|--|
| OpenSearch -<br>DataStream 和表 API | -          | flink-connector-op<br>ensearch, 1.2.0-1.18         | flink-connector-op<br>ensearch, 1.2.0-1.19         |
| OpenSearch - SQL                  | -          | flink-sql-connector-<br>opensearch, 1.2.0-1.1<br>8 | flink-sql-connector-<br>opensearch, 1.2.0-1.1<br>9 |

若要進一步了解適用於 Apache Flink 的 Amazon 受管服務中的連接器，請參閱：

- [DataStream API 連接器](#)
- [表格 API 連接器](#)

## 在 Apache Flink 的受管理服務中實作容錯能力

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 up-to-date 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](#)。

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，並使用應用程式進行設定 [ApplicationRestoreConfiguration](#)，則服務會在應用程式更新期間，或在與服務相關的調整或維護期間提供僅一次的處理語意。

## 在 Apache Flink 的受管理服務中設定檢查點

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`: 真
- `CheckpointInterval`: 60000
- `MinPauseBetweenCheckpoints`: 5000

如果 `ConfigurationType` 設定為 `DEFAULT`，則會使用前面的值，即使使用，或透過在應用程式碼中設定值設定為其他值。AWS Command Line Interface

#### 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。

## 檢閱檢查點範例 API

本節包括設定應用程式檢查點的API動作要求範例。若要取得有關如何使用JSON檔案進API行動作輸入的資訊，請參閱 [〈〉 適用於阿帕奇 Flink API 範例程式碼的託管服務](#)。

### 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_19",
  "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_19",
  "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

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。如需有關 Apache Flink 儲存點的資訊，請參閱 Apache Flink 文件中的[儲存點](#)。使用快照，您可以從應用程式狀態的特定快照集重新啟動應用程式。

### 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。拍攝頻繁的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 Apache Flink 的受管理服務中，您可以使用下列API動作來管理快照集：

- [CreateApplicationSnapshot](#)
- [DeleteApplicationSnapshot](#)
- [DescribeApplicationSnapshot](#)
- [ListApplicationSnapshots](#)

如需每個應用程式的快照數目限制，請參閱[適用於 Apache Flink 和工作室筆記本配額的託管服務](#)。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：

- [管理自動建立快照](#)
- [從包含不相容狀態資料的快照還原](#)
- [檢閱快照API範例](#)

## 管理自動建立快照

如果 `SnapshotsEnabled` 在應用程式的 `true` 中設定 [ApplicationSnapshotConfiguration](#) 為，Apache Flink 的受管理服務會在應用程式更新、調整或停止時自動建立並使用快照，以提供只有一次的處理語意。

**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](#) 動作查看快照集。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](#) 作手動刪除快照，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 `true` 使用 [UpdateApplication](#) 動作 [FlinkRunConfiguration](#) 將的 `AllowNonRestoredState` 參數設定為。

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。如需有關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 two-phase-commit 接收器，我們建議使用系統快照 (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，建議您將[StartApplication](#)動作的 ApplicationRestoreType 參數設定為，以略過從快照還原。SKIP\_RESTORE\_FROM\_SNAPSHOT

如需 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](#)。

## 檢閱快照API範例

本節包括將快照與應用程式搭配使用的API動作要求範例。若要取得有關如何使用JSON檔案進API行動作輸入的資訊，請參閱 [〈〉 適用於阿帕奇 Flink API 範例程式碼的託管服務](#)。

### 啟用應用程式的快照

[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
}
```

## 列出應用程式快照的詳細資訊

[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,
}
```

## 使用具名的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"  
    }  
  }  
}
```

## 為 Apache Flink 使用就地版本升級

透過適用於 Apache Flink 的就地版本升級，您可以針對ARN跨 Apache Flink 版本的單一應用程式保留可追蹤性。這包括快照、記錄、指標、標籤、Flink 組態、增加資源限制VPCs等等。

您可以針對 Apache Flink 執行就地版本升級，將現有應用程式升級到適用於 Apache Flink 的 Amazon 受管服務中的新 Flink 版本。若要執行此工作，您可以使用 AWS CLI AWS CloudFormation AWS SDK、或 AWS Management Console。

**Note**

您不能將 Apache Flink 的就地版本升級與 Amazon Flink 工作室的管理服務一起使用。

本主題包含下列章節：

- [使用 Apache Flink 的就地版本升級來升級應用程式](#)
- [將您的應用程式升級到新的 Apache Flink 版本](#)
- [還原應用程式升級](#)
- [應用程式升級的一般最佳作法和建議](#)
- [應用程式升級時的注意事項和已知](#)

## 使用 Apache Flink 的就地版本升級來升級應用程式

在開始之前，我們建議您觀看此視頻：[就地版本升級](#)。

若要針對 Apache Flink 執行就地版本升級，您可以使用 AWS CLI、AWS CloudFormation、AWS SDK、或 AWS Management Console。您可以在 READY 或 RUNNING 狀態下搭配 Apache Flink 管理服務使用的任何現有應用程式使用此功能。它會使用 UpdateApplication API 來新增變更 Flink 執行階段的能力。

### 升級前：更新您的 Apache Flink 應用程式

撰寫 Flink 應用程式時，您可 JAR 以將它們及其相依性捆綁到應用程式中，JAR 然後上傳到 Amazon S3 儲存貯體。從那裡，適用於 Apache Flink 的 Amazon 受管服務會在您選擇的新 Flink 執行階段中執行任務。您可能必須更新應用程式，才能與要升級至的 Flink 執行階段相容。Flink 版本之間可能存在不一致的情況，導致版本升級失敗。最常見的是，這將與來源（入口）或目的地（接收器，出口）和 Scala 依賴關係的連接器。Apache Flink 的管理服務中的 Flink 1.15 及更高版本是不可知的，並且您 JAR 必須包含您打算使用的 Scala 版本。

### 更新您的應用程式

1. 閱讀 Flink 社區關於使用狀態升級應程序的建議。請參閱 [升級應用程式和 Flink 版本](#)。
2. 閱讀了解問題和限制的列表。請參閱 [應用程式升級時的注意事項和已知](#)。
3. 更新您的相依性並在本機測試應用程式。這些依賴關係通常是：

1. Flink 執行階段和API。
2. 建議用於新 Flink 執行階段的連接器。您可以在要更新到的特定運行時的[發布版本](#)中找到這些內容。
3. 斯卡拉-阿帕奇 Flink 是與標準不可知論者開始，包括 Flink 1.15。您必須包含要在應用程序中使用使用的 Scala 依賴關係JAR。
4. JAR在 zip 檔案上建立新的應用程式，並將其上傳到 Amazon S3。我們建議您使用與之前的 JAR / zipfile 不同的名稱。如果您需要回滾，則將使用此信息。
5. 如果您正在執行可設定狀態的應用程式，我們強烈建議您建立目前應用程式的快照。如果在升級期間或之後遇到問題，這可讓您以狀態方式回復。

## 將您的應用程式升級到新的 Apache Flink 版本

您可以使用動作升級 Flink 應用程[UpdateApplication](#)式。

您可以透過多種方式呼叫：`UpdateApplicationAPI`

- 使用上的現有「組態」工作流程 AWS Management Console。
  - 前往您的應用程式頁面 AWS Management Console。
  - 選擇設定。
  - 選取新的執行階段和您要從其開始的快照，也稱為還原組態。使用最新設定做為還原設定，從最新的快照啟動應用程式。指向亞馬Amazon S3 上新升級的應用程式 JAR /zip。
- 使用更新應用 AWS CLI [程式動作](#)。
- 使用 AWS CloudFormation ( CFN )。
  - 更新[RuntimeEnvironment](#)欄位。之前，AWS CloudFormation 刪除了該應用程序並創建一個新應用程序，從而導致您的快照和其他應用程序歷史記錄丟失。現在 AWS CloudFormation 更新你 `RuntimeEnvironment` 的地方，不會刪除您的應用程序。
- 使用 AWS SDK。
  - 請參閱SDK文件以瞭解您選擇的程式設計語言。請參閱[UpdateApplication](#)。

您可以在應用程式RUNNING處於狀態或應用程式處於狀態停止時執行READY升級。適用於 Apache Flink 的 Amazon 受管服務會驗證原始執行階段版本與目標執行階段版本之間的相容性。此相容性檢查會在您處於狀態執行[UpdateApplication](#)時執行，[StartApplication](#)如果在RUNNING狀態下進行升級，則會在READY下一次執行。

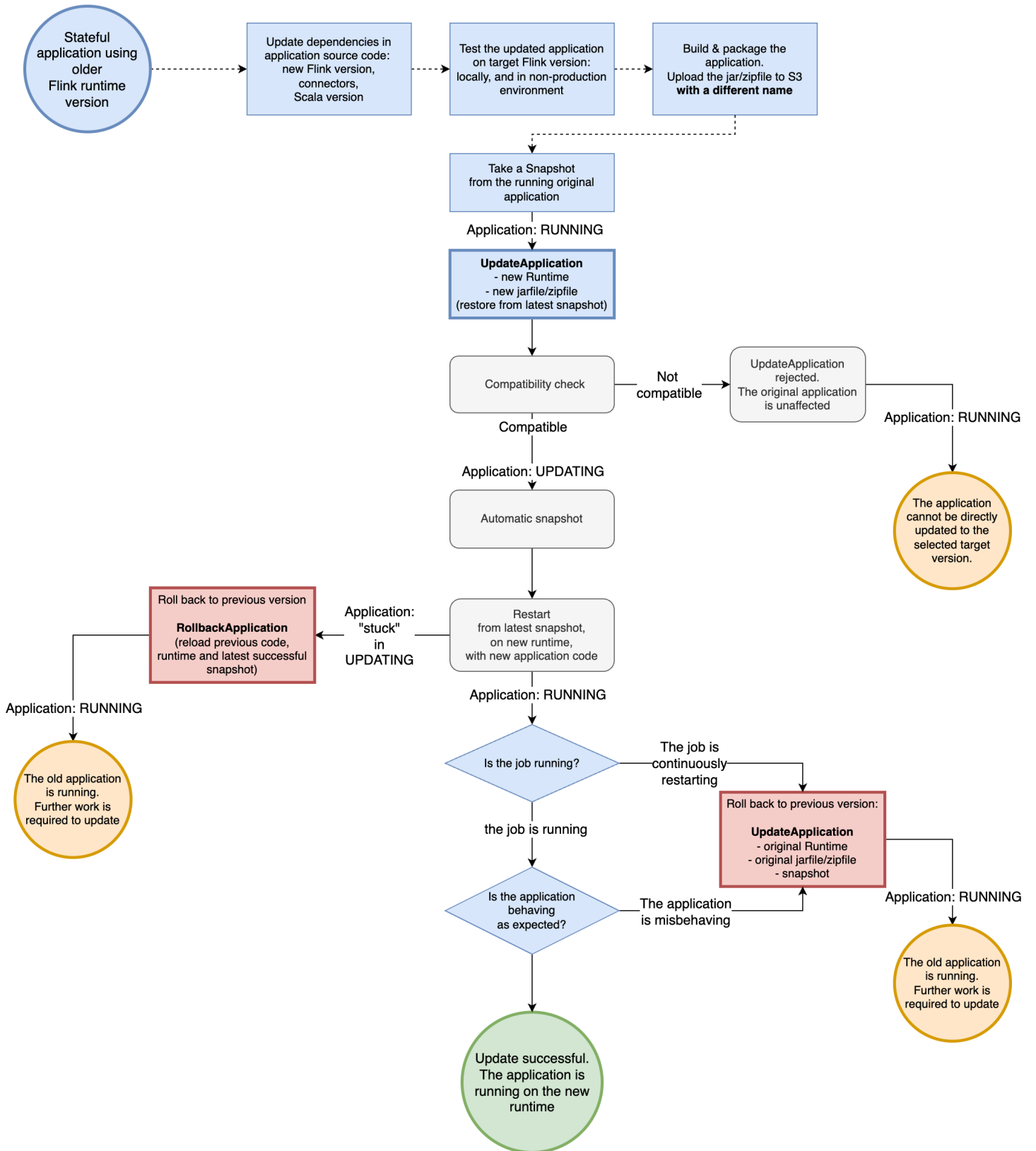
## 升級RUNNING處於狀態的應用程式

下列範例顯示使用 AWS CLI 並從最新快照啟動升級的應用程式，UpgradeTest將位於美國東部 (維吉尼亞北部) 的RUNNING狀態名為 Flink 1.18 的應用程式升級為 Flink 1.18。

```
aws --region us-east-1 kinesisanalyticstv2 update-application \  
--application-name UpgradeTest --runtime-environment-update "FLINK-1_18" \  
--application-configuration-update '{"ApplicationCodeConfigurationUpdate": \  
'{"CodeContentUpdate": {"S3ContentLocationUpdate": \  
'{"FileKeyUpdate": "flink_1_18_app.jar"}}}' \  
--run-configuration-update '{"ApplicationRestoreConfiguration": \  
'{"ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"}}' \  
--current-application-version-id ${current_application_version}
```

- 如果您已啟用服務快照並希望繼續最新快照中的應用程式，適用於 Apache Flink 的 Amazon 受管服務會驗證目前RUNNING應用程式的執行階段與選取的目標執行階段是否相容。
- 如果您已指定要從中繼續目標執行階段的快照，適用於 Apache Flink 的 Amazon 受管服務會驗證目標執行階段與指定的快照相容。如果相容性檢查失敗，您的更新要求會遭到拒絕，且您的應用程式會在RUNNING狀態下保持不變。
- 如果您選擇在沒有快照的情況下啟動應用程式，適用於 Apache Flink 的 Amazon 受管服務不會執行任何相容性檢查。
- 如果升級的應用程式失敗或卡在轉移UPDATING狀態，請按照[還原應用程式升級](#)本節中的指示返回健康狀態。

## 執行狀態應用程式的程序流程



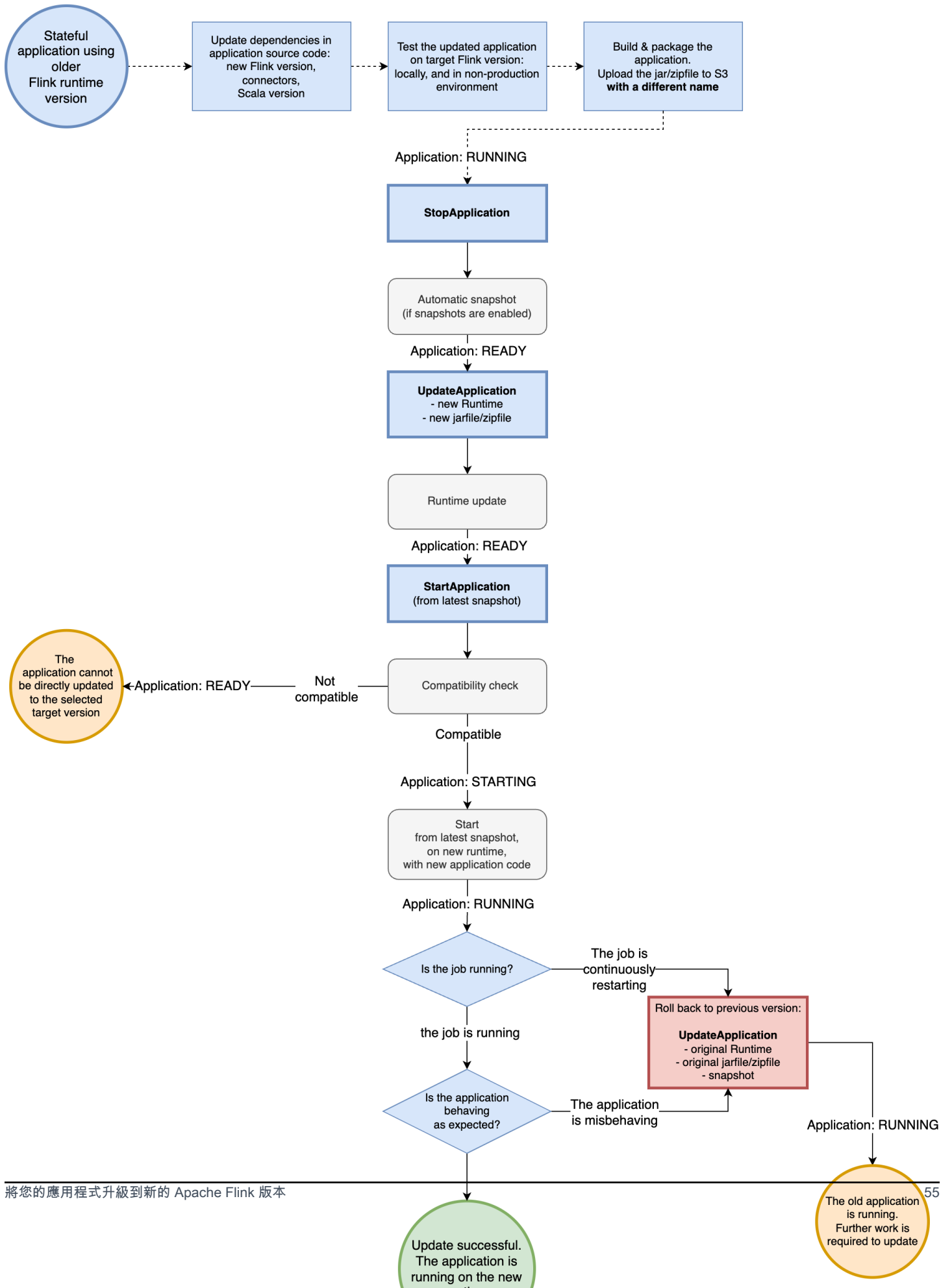
## 升級READY處於狀態的應用程式

下列範例顯示使用 `UpgradeTest` 將美國東部 (維吉尼亞北部) READY 狀態名為 Flink 1.18 的應用程式升級為 `AWS CLI` 沒有指定的快照來啟動應用程式，因為應用程式未執行。您可以在發出啟動應用程式要求時指定快照集。

```
aws --region us-east-1 kinesisanalyticstv2 update-application \
--application-name UpgradeTest --runtime-environment-update "FLINK-1_18" \
--application-configuration-update '{"ApplicationCodeConfigurationUpdate": '\
'{"CodeContentUpdate": {"S3ContentLocationUpdate": '\
'{"FileKeyUpdate": "flink_1_18_app.jar"}}}' \
--current-application-version-id ${current_application_version}
```

- 您可以將READY處於狀態的應用程式執行階段更新為任何 Flink 版本。適用於 Apache Flink 的 Amazon 受管服務不會執行任何檢查，直到您啟動應用程式為止。
- 適用於 Apache Flink 的 Amazon 受管服務只會針對您選取用來啟動應用程式的快照執行相容性檢查。這些是遵循 [Flink 相容性表格的基本相容性檢查](#)。他們只會檢查拍攝快照時使用的 Flink 版本以及您目標的 Flink 版本。如果所選快照的 Flink 執行階段與應用程式的新執行階段不相容，則啟動要求可能會遭到拒絕。

## 就緒狀態應用程式的處理流程



## 還原應用程式升級

如果您的應用程式發生問題，或發現 Flink 版本之間的應用程式程式碼不一致，您可以使用 AWS CLI、AWS CloudFormation、AWS SDK、或 AWS Management Console 下列範例顯示在不同失敗案例中回復的外觀。

執行階段升級成功，應用程式 **RUNNING** 處於狀態，但工作失敗並持續重新啟動

假設您嘗試將名為 TestApplication Flink 1.15 的可設定狀態應用程式升級至美國東部 (維吉尼亞北部) 的 Flink 1.18。但是，即使應用程式處於 **RUNNING** 狀態，升級後的 Flink 1.18 應用程式仍無法啟動或持續重新啟動。這是常見的失敗案例。為避免進一步的停機時間，我們建議您立即將應用程式復原至先前執行的版本 (Flink 1.15)，並在稍後診斷問題。

若要將應用程式復原至先前執行的版本，請使用 [回溯應用程式](#) AWS CLI 命令或動作 [RollbackApplication](#) API。此 API 動作會復原您所做的變更，導致最新版本。然後它使用最新的成功快照重新啟動您的應用程式。

我們強烈建議您在嘗試升級之前，先使用現有的應用程式拍攝快照。這將有助於避免數據丟失或必須重新處理數據。

在這個失敗的情況下，AWS CloudFormation 將不會回滾應用程序為您。您必須更新 CloudFormation 範本，以指向上一個執行階段和先前的程式碼，才能強制 CloudFormation 更新應用程式。否則，CloudFormation 假設您的應用程式在轉換到 **RUNNING** 狀態時已更新。

## 回滾卡住的應用程序 **UPDATING**

如果您的應用程式在嘗試升級後卡在 **UPDATING** 或 **AUTOSCALING** 狀態中，適用於 Apache Flink 的 Amazon 受管服務會提供 [回復應用程式](#) AWS CLI 命令，或可將應用程式回復到卡住 **UPDATING** 或狀態之前的版本的 [RollbackApplications](#) API 動作。AUTOSCALING 這會 API 復原您所做的變更，導致應用程式卡在 **UPDATING** 或 **AUTOSCALING** 傳遞狀態。

## 應用程式升級的一般最佳作法和建議

- 在嘗試生產升級之前，請在非生產環境上測試新的工作/執行階段 (不含狀態)。
- 請考慮先使用非生產應用程式測試可設定狀態升級。
- 請確定您的新工作圖表與用來啟動升級應用程式的快照具有相容的狀態。
  - 確保存在運算符狀態中的類型保持不變。如果類型已變更，Apache Flink 就無法還原運算子狀態。



- 請確定IDs您使用該uid方法設定的運算子保持不變。阿帕奇 Flink 有一個強烈的建議，分配唯IDs一的運營商。如需詳細資訊，請參閱 Apache Flink 文件IDs中的[指派運算子](#)。

如果您沒有指派IDs給您的運算子，Flink 會自動產生這些運算子。在這種情況下，它們可能取決於程序結構，如果更改，可能會導致兼容性問題。Flink 使用運算子IDs來比對快照中的狀態與運算子。變更運算子會IDs導致應用程式未啟動，或儲存在快照集中的狀態會遭到捨棄，而新的運算子則啟動時沒有狀態。

- 不要更改用於存儲密鑰狀態的密鑰。
- 不要修改有狀態運算符的輸入類型，如窗口或連接。這會隱含地變更運算子內部狀態的類型，造成狀態不相容。

## 應用程式升級時的注意事項和已知

### 狀態相容性的已知限制

- 如果您使用的是表API，Apache Flink 不保證 Flink 版本之間的狀態兼容性。如需詳細資訊，請參閱 Apache Flink 文件中的可設定[狀態升級和演進](#)。
- FLINK 1.6 狀態與 Flink 1.18 不相容。如果您嘗試使用狀態從 1.6 升級到 1.18 及更高版本，則API會拒絕您的請求。您可以升級至 1.8、1.11、1.13 和 1.15 並建立快照，然後升級至 1.18 及更新版本。如需詳細資訊，請參閱 Apache [Flink 文件中的 < 升級應用程式和 Flink 版本 >](#)。

### Flink Kinesis 連接器的已知問題

- 如果您使用 Flink 1.11 或更早版本，並且使用amazon-kinesis-connector-flink連接器進行 Enhanced-fan-out (EFO) 支援，則必須採取額外步驟才能將狀態升級至 Flink 1.13 或更新版本。這是因為連接器的套件名稱有所變更。如需詳細資訊，請參閱[amazon-kinesis-connector-flink](#)。

Flink 1.11 及更早版本的amazon-kinesis-connector-flink連接器使用封裝software.amazon.kinesis，而 Flink 1.13 及更新版本的 Kinesis 連接器則使用org.apache.flink.streaming.connectors.kinesis使用此工具支持您的遷移：[amazon-kinesis-connector-flink-狀態遷移器](#)。

- 如果您使用 Flink 1.13 或更早版本FlinkKinesisProducer並升級到 Flink 1.15 或更高版本，若要進行可狀態升級，您必須繼續FlinkKinesisProducer在 Flink 1.15 或更新版本中使用，而不是更新版本。KinesisStreamsSink但是，如果您的接收器上已經有自定義uid設置，則應該可以切換到，KinesisStreamsSink因為FlinkKinesisProducer不保持狀態。Flink 會將其視為相同的運算子，因為uid已設定自訂。

## 用斯卡拉編寫的 Flink 應用程式

- 從 Flink 1.15 開始，阿帕奇 Flink 不包括斯卡拉在運行時。升級到 Flink 1.15 或更高版本時，您必須在代碼 JAR /zip 中包含要使用的 Scala 版本和其他 Scala 依賴項。如需詳細資訊，請參閱[阿帕奇 Flink 1.15.2 版本的 Amazon 管理服務](#)。
- 如果您的應用程式使用斯卡拉，並且您正在將其從 Flink 1.11 或更早版本（斯卡拉 2.11）升級到 Flink 1.13（斯卡拉 2.12），請確保您的代碼使用斯卡拉 2.12。否則，您的 Flink 1.13 應用程式可能無法在 Flink 1.13 運行時找到斯卡拉 2.11 類。

## 降級 Flink 應用時要考慮的事項

- 可以降級 Flink 應用程式，但僅限於先前使用舊版 Flink 執行應用程式的情況。對於可狀態升級 Apache Flink 的受管理服務，將需要使用與相符或較早版本拍攝的快照進行降級
- 如果您將運行時從 Flink 1.13 或更高版本更新到 Flink 1.11 或更早版本，並且如果您的應用程式使用 HashMap 狀態後端，則您的應用程式將不斷失敗。

## 在 Apache Flink 的受管理服務中實作應用程式調整

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 如何排程 parallel 工作執行個體的相關資訊，請參閱 Apache Flink 文件中的[平行執行](#)。

### 主題

- [設定應用程式平行處理和 ParallelismPer KPU](#)
- [分配 Kinesis 處理單元](#)
- [更新應用程式的平行處理](#)
- [在適用於 Apache Flink 的受管理服務中使用自動調整](#)
- [maxParallelism 考量](#)

## 設定應用程式平行處理和 ParallelismPer KPU

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。除非在應用程式的程式碼中覆寫，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**— 使用此內容可設定每個應用程式的 Kinesis 處理單元 (KPU) 可排程的 parallel 工作數目。預設值為 1，最大值為 8。對於具有封鎖作業的應用程式 (例如 I/O)，較高的值 ParallelismPerKPU 會導致 KPU 資源的完全使用率。

#### Note

的限制等 Parallelism 於限制的 ParallelismPerKPU 乘以 KPU (預設值為 64)。可以通過請求提高 KPU 限制來增加限制。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」。

如需有關為特定運算子設定工作平行處理原則的資訊，請參閱 Apache Flink 文件中的 [設定平行處理原則：運算子](#)。

## 分配 Kinesis 處理單元

適用於 Apache Flink 的受管理服務將容量佈建為 KPU。單一記憶體可 KPU 為您提供 1 v CPU 和 4 GB 的記憶體。針對每個 KPU 配置，也會提供 50 GB 的執行中應用程式儲存空間。

Apache Flink 的受管理服務會計算 KPU 使用 Parallelism 和 ParallelismPerKPU 屬性執行應用程式所需的項目，如下所示：

```
Allocated KPU 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

額外收取的費用 KPU 是為了協調的目的。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原則之API動作的範例要求。如需有關如何將要求區塊與API動作搭配使用的更多範例和指示，請參閱[適用於阿帕奇 Flink API 範例程式碼的託管服務](#)。

[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_18",
  "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"
      }
    }
  }
}
}

```

## 在適用於 Apache Flink 的受管理服務中使用自動調整

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。預設會啟用自動擴展。Apache Flink 的受管理服務會監控應用程式的資源 (CPU) 使用情況，並隨之彈性地擴展應用程式的平行處理原則：

- 如果 CloudWatch 指標最containerCPUUtilization大值超過 75% 或更高，持續 15 分鐘，您的應用程式可擴展 (增加平行處理原則)。這表示ScaleUp當連續 15 個資料點且 1 分鐘期間等於或超過 75% 時，就會啟動動作。一個ScaleUp動作將您CurrentParallelism的應用程序的兩倍。ParallelismPerKPU未修改。因此，分配的數量KPU也會加倍。
- 當您的CPU使用量維持在百分之十以下六小時時，您的應用程式會縮減 (減少平行處理原則)。這表示ScaleDown當有 360 個連續資料點且 1 分鐘期間少於 10% 時，就會啟動動作。ScaleDown動作會將應用程式的平行度減半 (四捨五入)。ParallelismPerKPU沒有修改，分配的數量KPU也減半 (四捨五入)。

**Note**

可以參考containerCPUUtilization超過 1 分鐘的期間的最大值，以找出與用於 Scaling 動作的資料點的相關性，但不需要反映動作初始化時的確切時刻。

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KPU支付 Apache Flink 規定的受管理服務費用，這是應用程式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。
- 您的應用程式的預設限制為 64 KPU。如需詳細資訊，請參閱[適用於 Apache Flink 和工作室筆記本配額的託管服務](#)。
- 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parallelismPerKPU [UpdateApplication](#) 操作。如需有關設定應用程式平行處理原則設定的詳細資訊，請參閱[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 “適用於 Apache Flink 的受管理服務中的度量和維度”](#)。

## 實作自訂的自動調度

如果您想要對自動調度資源進行更精細的控制，或使用觸發指標以外的其他方式 `containerCPUUtilization`，您可以使用以下範例：

- [AutoScaling](#)

本範例說明如何使用與 Apache Flink 應用程式不同的指 CloudWatch 標來擴展 Apache Flink 應用程式的受管服務，包括來自 Amazon MSK 和 Amazon Kinesis Data Streams 的指標，用作來源或接收器。

如需詳細資訊，請參閱 [Apache Flink 的增強型監控和自動調整規模](#)。

## 實作排定的自動調度

如果您的工作負載隨著時間的推移遵循可預測的描述檔，您可能會偏好先發製擴展 Apache Flink 應用程式。這會在排定的時間調整您的應用程式，而不是根據量度反應調整。若要在一天中的固定時間設定擴展和縮減比例，您可以使用以下範例：

- [ScheduledScaling](#)

## maxParallelism 考量

Flink 工作可擴充的最大平行處理原則受到工作所有操作員的最小 `maxParallelism` 值限制。例如，如果您有一個只有來源和接收器的簡單工作，而來源的一個 `maxParallelism` 為 16，而接收器有 8，則應用程式無法擴充超過 8 的平行處理原則。

若要瞭解如何計算運算子 `maxParallelism` 的預設值以及如何覆寫預設值，請參閱在 Apache Flink 文件中 [設定最大平行程度](#)。

作為基本規則，請注意，如果您沒有定義 `maxParallelism` 任何運算符，並且您以小於或等於 128 的並行性啟動應用程序，則所有運算符都將具有 128 `maxParallelism` 的。

### Note

工作的最大平行處理原則是擴充應用程式保留狀態的平行處理原則上限。  
如果您修改 `maxParallelism` 現有應用程式，應用程式將無法從先前使用舊快照拍攝的快照重新啟動 `maxParallelism`。您只能重新啟動沒有快照的應用程式。

如果您打算將應用程式擴展到大於 128 的平行處理原則，則必須在應用程式 `maxParallelism` 中明確設定。

- 自動調度資源邏輯可防止將 Flink 工作擴展至超過工作最大平行處理原則的平行處理原則。
- 如果您使用自訂的自動調度資源或排程資源調整，請進行設定，使其不會超過工作的最大平行程度。
- 如果您手動將應用程式擴展到最大平行處理原則以外，應用程式將無法啟動。

## 將標籤新增至 Apache Flink 應用程式的受管理服務

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 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。

注意有關標記的以下資訊：

- 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](#)
- [新增或更新現有應用程式的標籤](#)
- [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"  
  }  
]
```

## 新增或更新現有應用程式的標籤

您可以使用動作將標記新增至應用程式 [TagResource](#) 式。您無法使用動作將標記新增至應用程式 [UpdateApplication](#) 式。

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

若要列出現有的標籤，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 CloudFormation 配 Apache Flink 的受管理服務搭配使用

下列練習說明如何啟動使用相同堆疊中的 Lambda 函數建立的 Flink 應 AWS CloudFormation 應用程式。

### 開始之前

開始本練習之前，請遵循使用 [...](#) 應用程式建立 Flink 應用 AWS CloudFormation 程式的步驟。AWS KinesisAnalytics

### 編寫一個 Lambda 函數

[要在創建或更新後啟動 Flink 應用程序，我們使用運動分析 sv2 啟動應用程序。](#) 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 執行期。

```
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 {}:\n{}'.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 會處理傳入 AWS CloudFormation 事件、篩選掉除外的所有內容 Update，Create 並取得應用程式狀態，並在狀態為時啟動它 READY。若要取得應用程式狀態，您必須建立 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。此角色使用預設受管理的策略，但您可能想要將其縮小為使用自訂策略。

```
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。

## 叫用 Lambda 函數

現在剩下要做的就是調用 Lambda 函數。您可以使用 [自訂資源](#) 來執行此操作。

```

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。

## 檢閱延伸範例

下列範例是稍微延伸的先前步驟版本，並透過 [範本參數](#) 進行額外的 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:
                - kinesisanalytics.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonKinesisFullAccess
        - arn:aws:iam::aws:policy/AmazonS3FullAccess
      Path: /
  InputKinesisStream:
    Type: AWS::Kinesis::Stream
    Properties:
      ShardCount: 1
  OutputKinesisStream:
    Type: AWS::Kinesis::Stream
    Properties:
      ShardCount: 1
```

```
TestFlinkApplication:
  Type: 'AWS::kinesisanalyticsv2::Application'
  Properties:
    ApplicationName: 'CFNTestFlinkApplication'
    ApplicationDescription: 'Test Flink Application'
    RuntimeEnvironment: 'FLINK-1_18'
    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 kinesisanalyticv2 API to start an application.
        client_kda = boto3.client('kinesisanalyticv2',
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"
  }
]

```

使用您的特定需求取代 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。

現在創建堆棧 ( 用您選擇REGION的區域替換 YOUR\_ , 例如 us-east-1 ) :

```
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
```

您現在可以瀏覽至 <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 \( 第 1 部分 , 共 3 部分 \) 檢索任何 AWS 服務屬性的四種方法。](#)
- [逐步解說：查找 Amazon 機器映像IDs。](#)

## 搭配 Apache Flink 管理服務使用 Apache Flink 儀表板

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空位。
- 作業資訊，包括正在執行、已完成、已取消和失敗的作業。

如需 Apache Flink 任務管理員、任務空位和作業的相關資訊，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 Apache Flink 的受管理服務主控台存取應用程式的 Apache Flink 儀表板，或使URL用 . CLI

## 使用適用於 Apache Flink 的受管理服務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

當您從 Apache Flink 的受管理服務主控台開啟儀表板時，主控台產生的有效期為 12 小時。URL

## 使用 Apache Flink 的受管理服務存取應用程式的 Apache Flink 儀表板 CLI

您可以使用 Apache Flink 的受管理服務 CLI 來產生存取您的應 URL 用程式儀表板。您產生 URL 的在指定的時間內有效。

### Note

如果您 URL 在三分鐘內不訪問生成的，它將不再有效。

您可以 URL 使用 [CreateApplicationPresignedUrl](#) 動作產生儀表板。您可以為動作指定下列參數值：

- 應用程式名稱
- 將有效的時間 (以秒 URL 為單位)
- 您指定 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。

### Note

如果您使用的是已取代的 Apache Flink 版本，建議您使用 Apache Flink 受管理服務中的 [為 Apache Flink 使用就地版本升級](#) 功能，將應用程式升級為最新受支援的 Flink 版本。

| 阿帕奇菲林克版 | 狀態-阿帕奇 Flink 的 Amazon 託管服務 | 狀態-阿帕奇 Flink 社區 | 連結   |
|---------|----------------------------|-----------------|--|
| 1.19.1  | 支援                         | 支援              | <a href="#">阿帕奇 Flink 1.19 Amazon 託管服務</a>                   |
| 1.18.1  | 支援                         | 支援              | <a href="#">阿帕奇 FLINK 1.18 Amazon 託管服務</a>                   |
| 1.15.2  | 支援                         | 不支援             | <a href="#">阿帕奇 Flink 1.15 的 Amazon 託管服務</a>                 |
| 1.13.1  | 支援                         | 不支援             | <a href="#">開始使用：快速</a>                                      |
| 1.11.1  | 棄用                         | 不支援             | <a href="#">適用於 Apache Flink 的受管理服務的舊版本資訊 (棄用二〇二四年十一月五日)</a> |
| 1.8.2   | 棄用                         | 不支援             | <a href="#">適用於 Apache Flink 的受管理服務的舊版本資訊 (棄用二〇二四年十一月五日)</a> |
| 1.6.2   | 棄用                         | 不支援             | <a href="#">適用於 Apache Flink 的受管理服務的舊版</a>                   |

| 阿帕奇菲林克版 | 狀態-阿帕奇 Flink 的 Amazon 託管服務 | 狀態-阿帕奇 Flink 社區 | 連結                                   |
|---------|----------------------------|-----------------|--------------------------------------|
|         |                            |                 | <a href="#">本資訊</a> ( 棄用二〇二四年十一月五日 ) |

## 主題

- [阿帕奇 Flink 1.19 Amazon 託管服務](#)
- [阿帕奇 FLINK 1.18 Amazon 託管服務](#)
- [阿帕奇 Flink 1.15 的 Amazon 託管服務](#)
- [適用於 Apache Flink 的受管理服務的舊版本資訊](#)

## 阿帕奇 Flink 1.19 Amazon 託管服務

阿帕奇 Flink 的託管服務現在支持阿帕奇 Flink 版本 1.19.1。本節將向您介紹與 Apache Flink 1.19.1 支援的 Apache Flink 支援的受管理服務所引入的主要新功能和變更。

### Note

如果您使用的是較早支援的 Apache Flink 版本，並且想要將現有的應用程式升級到 Apache Flink 1.19.1，您可以使用就地 Apache Flink 版本升級來執行此操作。如需詳細資訊，請參閱 [為 Apache Flink 使用就地版本升級](#)。透過就地版本升級，您可以 ARN 跨 Apache Flink 版本保留應用程式可追溯性，包括快照、記錄、指標、標籤、Flink 組態等。

## 支援的功能

Apache Flink 1.19.1 在中引入了改進功能 SQLAPI，例如具名參數、自訂來源平行處理原則，以及各種 Flink 運算子 TTLs 的不同狀態。

## 支援的功能和相關文件

| 支援的功能                                 | 描述  | 阿帕奇 Flink 文檔參考   |
|---------------------------------------|---|--|
| SQLAPI : Support TTLs 使用 SQL 提示設定不同狀態 | 使用者現在可以 TTL 在串流一般聯結和群組彙總上設定狀態。  | <a href="#">FLIP-373 : 使用提示設定不同的狀態 TTLs SQL</a>                      |
| SQLAPI : Support 函數和呼叫程序的命名參數         | 使用者現在可以在函數中使用命名參數，而不是依賴參數的順序。   | <a href="#">FLIP-378 : Support 函數和呼叫程序的命名參數</a>                      |
| SQLAPI : 設定來源的平行程度 SQL                | 使用者現在可以指定 SQL 來源的平行處理原則。  | <a href="#">FLIP-367 : Support 設定表格/來源的平行度 SQL</a>                   |
| SQLAPI: Support 階段作業視窗 TVF            | 用戶現在可以使用會話窗口表值函數。   | <a href="#">FLINK-24024 : Support 工作階段視窗 TVF</a>                     |
| SQLAPI : 視窗 TVF 彙總支援變更記錄檔輸入           | 使用者現在可以對變更記錄檔輸入執行視窗彙總。  | <a href="#">FLINK-20281 : 視窗彙總支援變更記錄檔串流輸入</a>                        |
| Support Python                        | Flink 現在支持 Python 3.11，與 Python 3.10 相比，速度快 10-60%。如需詳細資訊，請參閱 <a href="#">Python 3.11 中的新增功能</a> 。                    | <a href="#">FLINK-33030 : 新增蟒蛇 3.11 支援</a>                           |
| 提供 TwoPhaseCommitting 接收器的指標          | 用戶可以在兩個階段提交接收器中查看有關提交者狀態的統計信息。  | <a href="#">FLIP-371 : 為中的提交者建立提交者提供初始化內容 TwoPhaseCommittingSink</a> |
| 追蹤工作重新啟動和檢查點的記錄器                      | 使用者現在可以監控有關檢查點持續時間和回復趨勢的追蹤。在適用於 Apache Flink 的 Amazon 受管服務中，我們預設會啟用 SLF4j 追蹤記錄器，因此使用者可以透過應用程式記錄來監控檢查點和任務追蹤。CloudWatch | <a href="#">FLIP-384 : 引入 TraceReporter 並使用它來建立檢查點和復原追蹤</a>          |

**Note**

您可以透過提交[支援案例](#)來選擇使用下列功能：

## 選擇加入功能和相關文件

| 選擇加入功能                    | 描述  | 阿帕奇 Flink 文檔參考                                       |
|---------------------------|---|--|
| Support 在源處理積壓時使用更大的檢查點間隔 | 這是一項選擇加入功能，因為使用者必須針對其特定工作需求調整組態。  | <a href="#">FLIP-309：當來源處理積壓時，Support 使用較大的檢查點間隔</a> |
| 將系統輸出和系統重定向到 Java 日誌      | 這是一項選擇加入功能。在適用於 Apache Flink 的 Amazon 受管服務上，預設行為是忽略來自 System.out 和 System.err 的輸出，因為生產中的最佳實務是使用原生 Java 記錄器。 | <a href="#">FLIP-390：Support 系統輸出並錯誤重新導向至LOG或捨棄</a>  |

如需阿帕奇 Flink 1.19.1 版本文件，請參閱[阿帕奇 Flink](#) 文件 v1.19.1。

## 阿帕奇 Flink 1.19.1 Amazon 託管服務的變化

### 記錄追蹤報告程式預設為啟用

Apache Flink 1.19.1 引入了檢查點和復原追蹤，讓使用者能夠更妥善地偵錯檢查點和工作復原問題。在適用於 Apache Flink 的 Amazon 受管服務中，這些追蹤會登入 CloudWatch 日誌串流，讓使用者分解任務初始化所花費的時間，並記錄檢查點的歷史大小。

### 默認重啟策略現在是指數延遲

在 Apache Flink 1.19.1 中，指數延遲重新啟動策略有重大改進。從 Flink 1.19.1 開始，在 Amazon 適用於 Apache Flink 的受管服務中，Flink 任務預設使用指數延遲重新啟動策略。這表示使用者工作可以更快地從暫時性錯誤中復原，但如果工作重新啟動持續存在，則不會超載外部系統



## 元件

| 元件   | 版本  |
|--|---|
| Java   | 11 (建議使用)   |
| Python   | 3.11  |
| Kinesis Data Analytics Flink 執行階段 ( ) aws-kinesisanalytics-runtime | 1.2.0   |
| 連接器  | 如需可用連接器的相關資訊，請參閱 <a href="#">Apache Flink 連接器</a> 。                   |
| <a href="#">Apache Beam (僅限於 Beam 應用程式)</a>                        | Flink 1.19 沒有兼容的阿帕奇 FLINK 亞軍。如需詳細資訊，請參閱 <a href="#">Flink 版本相容性</a> 。 |

## 已知問題

### 阿帕奇梁

目前在阿帕奇梁中沒有兼容的阿帕奇 Flink 1.19 的 Flink 亞軍。如需詳細資訊，請參閱 [Flink 版本相容性](#)。

### 阿帕奇 Flink 工作室 Amazon 託管服務

Studio 使用 Apache 齊柏林飛艇筆記本提供單一介面開發經驗，用於開發、除錯程式碼和執行 Apache Flink 串流處理應用程式。齊柏林飛艇的 Flink 解譯器需要升級才能支援 Flink 1.19。這項工作安排在齊柏林飛艇社區中，我們將在完成後更新這些註釋。您可以繼續使用 Flink 1.15 與 Amazon 管理服務阿帕奇 Flink 工作室。如需詳細資訊，請參閱 [建立 Studio 筆記本](#)。

## 阿帕奇 FLINK 1.18 Amazon 託管服務

阿帕奇 Flink 的託管服務現在支持阿帕奇 Flink 版本 1.18.1。瞭解關於 Apache Flink 1.18.1 支援的 Apache Flink 支援的受管理服務所引入的主要新功能和變更。

**Note**

如果您使用的是較早支援的 Apache Flink 版本，並且想要將現有的應用程式升級到 Apache Flink 1.18.1，您可以使用就地 Apache Flink 版本升級來執行此操作。透過就地版本升級，您可以以ARN跨 Apache Flink 版本保留應用程式可追溯性，包括快照、記錄、指標、標籤、Flink 組態等。您可以在RUNNING和READY狀態中使用此功能。如需詳細資訊，請參閱 [為 Apache Flink 使用就地版本升級](#)。

| 支援的功能                                   | 描述   | 阿帕奇 Flink 文檔參考   |
|---|--|--|
| 開放搜索連接器                                 | 該連接器包括一個提供保 at-least-once 證的水槽。                    | <a href="#">github: 開放搜索連接器</a>  |
| Amazon DynamoDB 接器                      | 該連接器包括一個提供保 at-least-once 證的水槽。                    | <a href="#">Amazon DynamoDB 槽</a>                                      |
| MongoDB 器                               | 該連接器包括提供 at-least-once 保證的源和水槽。                    | <a href="#">MongoDB 器</a>  |
| 使用 Flink 規劃器分離蜂巢                        | 您可以直接使用 Hive 方言，而無需額外的JAR交換。                       | <a href="#">FLINK-26603：使用 Flink 規劃器將蜂巢分離</a>                          |
| 默認情況下WAL在 RocksDBWrite BatchWrapper 中禁用 | 這提供了更快的恢復時間。                                       | <a href="#">FLINK-32326：預設情況下，WAL在 R 中停用 RocksDBWrite BatchWrapper</a> |
| 改善啟用浮水印對齊時的浮水印聚合效能                      | 改善啟用浮水印對齊時的浮水印聚合效能，並新增相關基準測試。                      | <a href="#">FLINK-32524：浮水印聚合效能</a>                                    |
| 使水印對齊準備就緒以供生產使用                         | 消除大型工作超載的風險<br>JobManager                          | <a href="#">FLINK-32548：準備好水印對齊</a>                                    |
| 可配置 RateLimitingStrategy 的異步水槽          | RateLimitingStrategy 可讓您設定要擴展的項目、何時調整規模以及要擴展多少的決定。 | <a href="#">FLIP-242：引入可配置 RateLimitingStrategy 的異步接收器</a>             |

| 支援的功能         | 描述      | 阿帕奇 Flink 文檔參考                             |
|---------------|---------|--|
| 大量擷取資料表和資料欄統計 | 改善查詢效能。 | <a href="#">FLIP-247：批量獲取給定分區的表格和列統計信息</a> |

如需阿帕奇 Flink 1.18.1 版本文件，請參閱[阿帕奇 Flink 1.18.1 發行公告](#)。

## 阿帕奇 Flink 1.18 Amazon 管理服務的變化

### 阿卡替換為白馬

阿帕奇 Flink 取代阿卡與佩克在阿帕奇 FLINK 1.18。這項變更在 Apache Flink 1.18.1 及更新版本的 Apache Flink 受管理服務中完全支援。您不需要因為此變更而修改應用程式。如需詳細資訊，請參閱[FLINK-32468：以 Pekko 取代阿卡](#)。

### Support 線程模式下的 PyFlink 運行時執行

此 Apache Flink 變更為 Pyflink 執行階段架構 (處理程序模式) 引入了新的執行模式。處理模式現在可以在同一個線程中執行 Python 用戶定義函數，而不是單獨的進程。

## 元件

| 元件  | 版本  |
|---|---|
| Java  | 11 (建議使用)   |
| Scala   | 自 1.15 版本以來，Flink 是刻度不可知的。作為參考，MSFFlink 1.18 已經對斯卡拉 3.3 ( ) LTS進行了驗證。 |
| 阿帕奇 Flink 運行時的託管服務 ( ) aws-kinesisanalytics-runtime                                 | 1.2.0   |
| <a href="#">AWS Kinesis 連接器 ( flink-connector-kinesis )</a><br><a href="#">[來源]</a> | 4.2.0-1.18  |
| <a href="#">AWS Kinesis 連接器 (flink-connector-kinesis)</a><br><a href="#">[水槽]</a>   | 4.2.0-1.18  |

| 元件  | 版本  |
|---|---|
| <a href="#">Apache Beam (僅限於 Beam 應用程式)</a> | 早期和最多版本 2.75.0。如需詳細資訊，請參閱 <a href="#">Flink 版本相容性</a> 。 |

## 錯誤修正

### 阿帕奇菲林克 1.18.1 中的狀態壓縮

Apache Flink 為所有檢查點和保存點提供了可選的壓縮 ( 默認值：關閉 )。Apache Flink 在 Flink 1.18.1 中發現了一個錯誤，其中啟用快照壓縮時無法正確恢復操作員狀態。這可能會導致資料遺失或無法從檢查點還原。如需詳細資訊，請參閱 [FLINK-34063：啟用快照壓縮時，重新調整來源運算子會導致部分分割遺失](#)。

為了解決這個問題，適用於 Apache Flink 的 Amazon 受管服務已反向移植將包含在 future 版本的 Apache Flink 中的修正程式。有關更多信息，請參閱 [github：始終刷新壓縮緩衝區](#)。

## 已知問題

### 阿帕奇 Flink 工作室 Amazon 託管服務

Studio 使用 Apache 齊柏林飛艇筆記本提供單一介面開發經驗，用於開發、除錯程式碼和執行 Apache Flink 串流處理應用程式。齊柏林飛艇的 Flink 解譯器需要升級才能支援 Flink 1.18。這項工作安排在齊柏林飛艇社區中，我們將在完成後更新這些註釋。您可以繼續使用 Flink 1.15 與 Amazon 管理服務阿帕奇 Flink 工作室。如需詳細資訊，請參閱 [建立 Studio 筆記本](#)。

## 阿帕奇 Flink 1.15 的 Amazon 託管服務

阿帕奇 Flink 的託管服務支持阿帕奇 1.15.2 中的以下新功能：

| 功能     | 描述  | 阿帕奇FLIP參考                        |
|--------|---|----------------------------------|
| 非同步接收器 | 建置非同步目的地的 AWS 貢獻架構，可讓開發人員以前不到一半的努力來建置自訂 AWS 連接器。如需詳細資訊，請參閱 <a href="#">通用非同步基本接收器</a> 。 | <a href="#">FLIP-171：非同步</a> 水槽。 |

| 功能                        | 描述  | 阿帕奇FLIP參考   |
|---------------------------|---|---|
| Kinesis Data Firehose 接收器 | AWS 使用異步框架貢獻了一個新的 Amazon Kinesis Firehose 接收器。  | <a href="#">Amazon Kinesis Data Firehose 接收器</a> 。                          |
| 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。   | <a href="#">FLIP-34：使用儲存點終止/暫停 Job</a> 。                                    |
| Scala 解耦                  | 用戶現在可以利用任何API斯卡拉版本的 Java，包括斯卡拉 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。   | <a href="#">FLIP-28：製作無火石工作台標準的長期目標</a> 。                                   |
| Scala                     | 請參閱上面的 Scala 解耦   | <a href="#">FLIP-28：製作無火石工作台標準的長期目標</a> 。                                   |
| 統一的連接器指標                  | Flink 針對作業、任務和運算子擁有 <a href="#">已定義的標準指標</a> 。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 。 | <a href="#">FLIP-33: 標準化連接器度量</a> 和 <a href="#">FLIP-179: 公開標準化的操作員度量</a> 。 |
| 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。  | <a href="#">FLIP-147：任務完成後 Support 檢查點</a> 。                                |

# 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 [使用具有 Apache Flink 管理服務的工作室筆記本](#) 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 Apache Flink 1.15 版的受管理服務時，請確定您使用的是最新版本 EFO 連接器，也就是任何 1.15.3 版或更新版本的連接器。如需有關原因的詳細資訊，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

升級至適用於 Apache Flink 版本 1.15 的 Amazon 管理服務時，請確保您使用的是最新的卡夫卡連接器。APIs 阿帕奇 Flink 已經棄用 [FlinkKafkaConsumer](#)，[FlinkKafkaProducer](#) 這些 APIs 卡夫卡水槽不能提交給卡夫卡 Flink 1.15。請確定您正在使用 [KafkaSource](#) 和 [KafkaSink](#)。

## 元件

| 元件  | 版本        |
|---|-----------|
| Java  | 11 (建議使用) |
| Scala   | 2.12      |
| 阿帕奇 Flink 運行時的託管服務 ( ) aws-kinesisanalytics-runtime | 1.2.0     |

| 元件   | 版本                           |
|--|------------------------------|
| <a href="#">AWS Kinesis 連接器 () flink-connector-kinesis</a> | 1.15.4                       |
| <a href="#">Apache Beam (僅限於 Beam 應用程式)</a>                | 2.33.0 , 帶有 Jackson 2.12.2 版 |

## 適用於 Apache Flink 的受管理服務的舊版本資訊

### Note

阿帕奇 Flink 版本 1.6 , 1.8 和 1.11 一直沒有得到阿帕奇 Flink 社區的支持超過三年。我們計劃於 2024 年 11 月 5 日在 Amazon 管理服務中淘汰這些版本。從這個日期開始，您將無法為這些 Flink 版本建立新的應用程式。您目前可以繼續執行現有的應用程式。您可以使用適用於 Apache Flink 的 Amazon 受管服務中的就地版本升級功能，以狀態方式升級應用程式。如需詳細資訊，請參閱。[為 Apache Flink 使用就地版本升級](#)

阿帕奇 Flink 的 1.15.2 版和 1.13.2 版本受到 Apache Flink 管理服務的支援，但不再受到 Apache Flink 社群的支援。

本主題包含下列章節：

- [使用阿帕奇 Flink Kinesis 串流連接器與先前的 Apache Flink 版本](#)
- [使用阿帕奇 FLINK 1.8.2 建置應用程式](#)
- [使用阿帕奇 FLINK 1.6.2 建置應用程式](#)
- [升級應用](#)
- [阿帕奇 FLINK 1.6.2 和 1.8.2 中的可用連接器](#)
- [開始使用：快速](#)
- [開始使用：Flink 1.11.1-棄用](#)
- [開始使用：Flink 1.8.2-棄用](#)
- [開始使用：Flink 1.6.2-棄用](#)
- [適用於 Apache Flink 的受管理服務的舊版 \(舊版\) 範例](#)

## 使用阿帕奇 Flink Kinesis 串流連接器與先前的 Apache Flink 版本

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。



## 使用阿帕奇 FLINK 1.8.2 建置應用程式

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      |
| 適用於 Flink 運行時的阿帕奇 Flink 的託管服務<br>( ) aws-kinesisanalytics-runtime | 1.0.1      |
| 適用於阿帕奇 Flink 連接器的受管理服務 ( ) aws-kinesisanalytics-flink             | 1.0.1      |
| Apache Maven  | 3.1        |

若要編譯使用 Apache Flink 1.8.2 版的應用程式，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 [Apache Flink 應用程式建立受管理服務](#)。

## 使用阿帕奇 FLINK 1.6.2 建置應用程式

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                     |
| 適用於 Flink 運行時的阿帕奇 Flink 的託管服務<br>( ) aws-kinesisanalytics-runtime | 1.0.1                     |
| 適用於阿帕奇 Flink 連接器的受管理服務 ( ) 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 執行期 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 [為 Apache Flink 應用程式建立受管理服務](#)。

## 升級應用

若要升級適用於 Apache Flink 應用程式的 Amazon 受管服務的 Apache Flink 版本，請使用現地 Apache Flink 版本升級功能 AWS CLI，使用軟體 AWS 開發套件或 AWS CloudFormation AWS Management Console 如需詳細資訊，請參閱 [為 Apache Flink 使用就地版本升級](#)。

您可以將此功能與任何現有的應用程式搭配使用於 Apache Flink 的 Amazon 受管服務 (READY 或 RUNNING 狀態)。

## 阿帕奇 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 \(1.6.2\)](#) 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 \(1.8.2\)](#) 中的 [連接器 \(1.8.2\)](#)。

### 開始使用：快速

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

- [適用於 Apache Flink 應用程式的受管理服務元件](#)
- [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](#)
- [下一步驟](#)
- [步驟 2：設置 AWS Command Line Interface \( AWS CLI \)](#)
- [第 3 步：為 Apache Flink 應用程序創建並運行託管服務](#)
- [步驟 4：清理 AWS 資源](#)
- [步驟 5：後續步驟](#)

### 適用於 Apache Flink 應用程式的受管理服務元件

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 執行期生成輸出。

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

- 執行期屬性：您可以使用執行期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 [將串流資料來源新增至 Apache Flink 的受管理服務](#)。

- **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[在 Apache Flink 的受管理服務中使用運算子轉換資料](#)。
- **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、Firehose 串流、Amazon S3 儲存貯體等。如需詳細資訊，請參閱[在 Apache Flink 的受管理服務中使用接收器寫入資料](#)。

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 [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

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 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。

當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。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。

### 建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

### 保護您的 AWS 帳戶根使用者

1. 選擇 Root 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 \(主控台\)](#)。

### 建立具有管理權限的使用者

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 身分中心中，將管理存取權授予使用者。

[若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。](#)

### 以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

如需使用 IAM 身分中心使用者 [登入的說明](#)，請參閱 [使用AWS 登入者指南中的登入 AWS 存取入口網站](#)。

## 指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「[建立權限集](#)」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「[新增群組](#)」。

## 授與程式設計存取權

如果使用者想要與 AWS 之外的 AWS Management Console 授與程式設計存取權的方式取決於正在存取的使用者類型。

若要授與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<br>(IAM Identity Center 中管理的使用者) | 使用臨時登入資料來簽署對 AWS CLI、AWS SDK 或 AWS API 的程式設計要求。 | <p>請依照您要使用的介面所提供的指示操作。</p> <ul style="list-style-type: none"> <li>• 如需詳細資訊 AWS CLI，請參閱 <a href="#">《使 AWS CLI 用AWS Command Line Interface者指南》</a> AWS IAM Identity Center中的〈配置使用〉。</li> <li>• 如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 <a href="#">IAM 身分中心身分驗證</a>。</li> </ul> |

| 哪個使用者需要程式設計存取權？ | 到   | By   |
|-----------------|---|--|
| IAM             | 使用臨時登入資料來簽署對 AWS CLI、AWS SDK 或 AWS API 的程式設計要求。         | 遵循《IAM <a href="#">使用者指南</a> 》中的 <a href="#">〈將臨時登入資料搭配 AWS 資源使用〉</a> 中的指示   |
| IAM             | (不建議使用)<br>使用長期認證簽署對 AWS CLI、AWS SDK 或 AWS API 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。 <ul style="list-style-type: none"> <li>• 如需相關資訊 AWS CLI，請參閱使用指南中的<a href="#">使用 IAM 使用者登入資料進行驗證</a>。AWS Command Line Interface</li> <li>• 對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的<a href="#">使用長期憑據進行身份驗證</a>。</li> <li>• 如需 AWS API，請參閱 IAM 使用者指南中的<a href="#">管理 IAM 使用者的存取金鑰</a>。</li> </ul> |

下一步驟

[設置 AWS Command Line Interface \( AWS CLI \)](#)

下一步驟

[步驟 2：設置 AWS Command Line Interface \( AWS CLI \)](#)

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

在此步驟中，您會下載並設定 AWS CLI 要與 Apache Flink 的受管理服務搭配使用。

**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 AWS CLI 安裝了，則可能需要升級才能獲得最新功能。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](#)。若要檢查的版本 AWS CLI，請執行下列命令：

```
aws --version
```

本自學課程中的練習需要以下 AWS CLI 版本或更高版本：

```
aws-cli/1.16.63
```

**若要設定 AWS CLI**

1. 下載和設定 AWS CLI。如需相關指示，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. 在檔案中新增管理員使用者的具名設定 AWS CLI config 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，請參閱中的[區域和端點 Amazon Web Services 一般參考](#)。

**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。

下一步驟

### [第 3 步：為 Apache Flink 應用程式創建並運行託管服務](#)

## 第 3 步：為 Apache Flink 應用程式創建並運行託管服務

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 資料串流](#)
- [將樣本記錄寫入輸入流](#)
- [下載並檢查阿帕奇 Flink 流 Java 代碼](#)
- [編譯應用程式程式碼](#)
- [上傳阿帕奇 Flink 流 Java 代碼](#)
- [建立並執行 Apache Flink 應用程式的受管理服務](#)
- [下一步驟](#)

### 建立兩個 Amazon Kinesis 資料串流

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)，請使用下列 Amazon Kinesis create-stream AWS CLI 指令。

```
$ 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
```

## 將樣本記錄寫入輸入流

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 \(Boto\)](#)。

- 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, kineses_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
```

## 下載並檢查阿帕奇 Flink 流 Java 代碼

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. 導覽至 `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。

如需執行期屬性的詳細資訊，請參閱[在 Apache Flink 的受管理服務中使用執行階段屬性](#)。

## 編譯應用程式程式碼

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) 的相關資訊，請參閱[滿足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。如果您使用建立應用程式 AWS CLI，請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
```

## 上傳阿帕奇 Flink 流 Java 代碼

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 Apache Flink 應用程式的受管理服務

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。使用建立應用程式時 AWS CLI，您需要分別建立這些資源。

## 主題

- [創建並運行應用程序 \(控制台\)](#)
- [創建並運行應用程序 \(AWS CLI\)](#)

## 創建並運行應用程序 (控制台)

依照以下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**。
  - 對於執行期，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**

## 編輯身分與存取權管

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 選擇政策。選擇主控台為您在上節所建立的 **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。使用您的帳戶 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                |
|---------------------------------|----------------------------------|------------------|
| <b>ProducerConfigProperties</b> | <b>flink.inputstream.initpos</b> | <b>LATEST</b>    |
| <b>ProducerConfigProperties</b> | <b>aws.region</b>                | <b>us-west-2</b> |



| 群組 ID                           | 金鑰                        | 值            |
|---------------------------------|---------------------------|--------------|
| <b>ProducerConfigProperties</b> | <b>AggregationEnabled</b> | <b>false</b> |

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。

## 創建並運行應用程序 ( AWS CLI )

在本節中，您可 AWS CLI 以使用建立和執行 Apache Flink 應用程式的受管理服務。適用於 Apache Flink 的受管理服務會使用指kinesisanalyticsv2 AWS CLI 令來建立 Apache Flink 應用程式的受管理服務，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`。使用您的帳戶 ID 取代 Amazon Resource Name (ARN) (`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，選擇角色、建立角色。
3. 在選取可信身分類型下，選擇 AWS 服務。在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)。

選擇下一步：許可。

4. 在連接許可政策頁面，選擇下一步：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，選擇許可標籤。
- b. 選擇連接政策。
- 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 \(主控台\)](#)。

為 Apache Flink 應用程式建立受管理的服務

1. 將下列 JSON 程式碼複製到名為 `create_request.json` 的檔案。使用您之前建立之角色的 ARN，取代範例角色 ARN。使用您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (*username*)。使用您的帳戶 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. 使用前述請求執行 [CreateApplication](#) 動作以建立應用程式：

```
aws kinesisanalyticstv2 create-application --cli-input-json file://
create_request.json
```

應用程式現在已建立。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 kinesisanalyticstv2 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 kinesisanalyticstv2 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 “在 Apache Flink 的受管理服務中設定應用程式登入”](#)。

## 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 kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

## 更新應用程式的程式碼

當您需要使用新版本的程式碼套件更新應用程式程式碼時，請使用[UpdateApplication](#) AWS 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。如需如何使用 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 資料串流”](#)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": "SAMPLEUehYngP87ex1nzYIGYgfhyvpDU"
        }
      }
    }
  }
}
```

**下一步驟****[步驟 4：清理 AWS 資源](#)****步驟 4：清理 AWS 資源**

本節包括清理在入門教學課程中建立的 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)



- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)
- [下一步驟](#)

刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

刪除您的 Kinesis 資料串流

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。

刪除您的 Amazon S3 對象和存儲桶

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。

刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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-we st-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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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。

### 下一步驟

#### [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：適用於 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。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。
- [AWS 適用於 Amazon MSK 的串流資料解決方案](#)：Amazon MSK 的 AWS 串流資料解決方案提供 AWS CloudFormation 範本，讓資料流經生產者、串流儲存、消費者和目的地。
- [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計程車公司的運營。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。
- [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。

## 開始使用：Flink 1.11.1-棄用

**Note**

阿帕奇 Flink 版本 1.6，1.8 和 1.11 一直沒有得到阿帕奇 Flink 社區的支持超過三年。我們計劃於 2024 年 11 月 5 日在 Amazon 管理服務中淘汰這些版本。從這個日期開始，您將無法為這些 Flink 版本建立新的應用程式。您目前可以繼續執行現有的應用程式。您可以使用適用於 Apache Flink 的 Amazon 受管服務中的就地版本升級功能，以狀態方式升級應用程式。如需詳細資訊，請參閱。[為 Apache Flink 使用就地版本升級](#)

本主題包含使用 Apache Flink 1.11.1 的[教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#)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

- [適用於 Apache Flink 應用程式的受管理服務元件](#)
- [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 \(AWS CLI\)](#)
- [第 3 步：為 Apache Flink 應用程序創建並運行託管服務](#)
- [步驟 4：清理 AWS 資源](#)
- [步驟 5：後續步驟](#)

## 適用於 Apache Flink 應用程式的受管理服務元件

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 執行期生成輸出。

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

- **執行期屬性**：您可以使用執行期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[將串流資料來源新增至 Apache Flink 的受管理服務](#)。
- **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[在 Apache Flink 的受管理服務中使用運算子轉換資料](#)。
- **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、Firehose 串流、Amazon S3 儲存貯體等。如需詳細資訊，請參閱[在 Apache Flink 的受管理服務中使用接收器寫入資料](#)。

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 [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

### 註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

### 若要註冊成為 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。

當您註冊一個時 AWS 帳戶，將創建 AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。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。

### 建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

### 保護您的 AWS 帳戶根使用者

1. 選擇 Root 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 \(主控台\)](#)。

### 建立具有管理權限的使用者

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 身分中心中，將管理存取權授予使用者。

[若要取得有關使用 IAM Identity Center 目錄做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。](#)

以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

如需使用 IAM 身分中心使用者[登入的說明](#)，請參閱[使用AWS 登入者指南中的登入 AWS 存取入口網站](#)。

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「[建立權限集](#)」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「[新增群組](#)」。

授與程式設計存取權

如果使用者想要與 AWS 之外的 AWS Management Console 授與程式設計存取權的方式取決於正在存取的使用者類型。

若要授與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<br>(IAM Identity Center 中管理的使用者) | 使用臨時登入資料來簽署對 AWS CLI、AWS SDK 或 AWS API 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。<br><br>• 如需詳細資訊 AWS CLI，請參閱 <a href="#">《使 AWS CLI 用AWS Command Line Interface者</a> |

| 哪個使用者需要程式設計存取權？ | 到   | By  |
|-----------------|---|---|
|                 |   | <p>指南》AWS IAM Identity Center中的〈配置使用〉。</p> <ul style="list-style-type: none"> <li>如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 <a href="#">IAM 身分中心身分驗證</a>。</li> </ul>   |
| IAM             | 使用臨時登入資料來簽署對 AWS CLI、AWS SDK 或 AWS API 的程式設計要求。         | 遵循《IAM <a href="#">使用者指南</a> 》中的〈 <a href="#">將臨時登入資料搭配 AWS 資源使用</a> 〉中的指示  |
| IAM             | (不建議使用)<br>使用長期認證簽署對 AWS CLI、AWS SDK 或 AWS API 的程式設計要求。 | <p>請依照您要使用的介面所提供的指示操作。</p> <ul style="list-style-type: none"> <li>如需相關資訊 AWS CLI，請參閱使用指南中的<a href="#">使用 IAM 使用者登入資料進行驗證</a>。AWS Command Line Interface</li> <li>對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的<a href="#">使用長期憑據進行身份驗證</a>。</li> <li>如需 AWS API，請參閱 IAM <a href="#">使用者指南中的管理 IAM 使用者的存取金鑰</a>。</li> </ul> |

## 下一步驟

### [設置 AWS Command Line Interface \( AWS CLI \)](#)

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

在此步驟中，您會下載並設定 AWS CLI 要與 Apache Flink 的受管理服務搭配使用。

**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 AWS CLI 安裝了，則可能需要升級才能獲得最新功能。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](#)。若要檢查的版本 AWS CLI，請執行下列命令：

```
aws --version
```

本自學課程中的練習需要以下 AWS CLI 版本或更高版本：

```
aws-cli/1.16.63
```

**若要設定 AWS CLI**

1. 下載和設定 AWS CLI。如需相關指示，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. 在檔案中新增管理員使用者的具名設定 AWS CLI config 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，請參閱中的[區域和端點 Amazon Web Services 一般參考](#)。



**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。

下一步驟

### [第 3 步：為 Apache Flink 應用程式創建並運行託管服務](#)

## 第 3 步：為 Apache Flink 應用程式創建並運行託管服務

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 資料串流](#)
- [將樣本記錄寫入輸入流](#)
- [下載並檢查阿帕奇 Flink 流 Java 代碼](#)
- [編譯應用程式程式碼](#)
- [上傳阿帕奇 Flink 流 Java 代碼](#)
- [建立並執行 Apache Flink 應用程式的受管理服務](#)
- [下一步驟](#)

### 建立兩個 Amazon Kinesis 資料串流

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)，請使用下列 Amazon Kinesis create-stream AWS CLI 指令。

```
$ 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
```

### 將樣本記錄寫入輸入流

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
```

## 下載並檢查阿帕奇 Flink 流 Java 代碼

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. 導覽至 `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。

如需執行期屬性的詳細資訊，請參閱[在 Apache Flink 的受管理服務中使用執行階段屬性](#)。

## 編譯應用程式程式碼

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) 的相關資訊，請參閱[滿足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。如果您使用建立應用程式 AWS CLI，請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
```

上傳阿帕奇 Flink 流 Java 代碼

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 Apache Flink 應用程式的受管理服務

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。使用建立應用程式時 AWS CLI，您需要分別建立這些資源。

主題

- [創建並運行應用程式 \(控制台\)](#)
- [創建並運行應用程式 \(AWS CLI\)](#)

## 創建並運行應用程式 (控制台)

依照以下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**。
  - 對於執行期，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**

## 編輯身分與存取權管

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。

2. 選擇政策。選擇主控台為您在上節所建立的 **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。使用您的帳戶 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. 在屬性下，為群組 ID 輸入 **ProducerConfigProperties**。
5. 輸入以下應用程式屬性和數值：



| 群組 ID                           | 金鑰                               | 值                |
|---------------------------------|----------------------------------|------------------|
| <b>ProducerConfigProperties</b> | <b>flink.inputstream.initpos</b> | <b>LATEST</b>    |
| <b>ProducerConfigProperties</b> | <b>aws.region</b>                | <b>us-west-2</b> |
| <b>ProducerConfigProperties</b> | <b>AggregationEnabled</b>        | <b>false</b>     |

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。

## 創建並運行應用程式 ( AWS CLI )

在本節中，您 AWS CLI 將使用建立並執行 Apache Flink 應用程式的受管理服務。Apache Flink 的受管理服務會使用此 `kinesisanalyticsv2` AWS CLI 命令來建立 Apache Flink 應用程式的受管理服務，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`。使用您的帳戶 ID 取代 Amazon Resource Name (ARN) (`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，選擇角色、建立角色。
3. 在選取可信身分類型下，選擇 AWS 服務。在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)。

選擇下一步：許可。

4. 在連接許可政策頁面，選擇下一步：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，選擇許可標籤。
- b. 選擇連接政策。
- 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 \(主控台\)](#)。

為 Apache Flink 應用程式建立受管理的服務

1. 將下列 JSON 程式碼複製到名為 `create_request.json` 的檔案。使用您之前建立之角色的 ARN，取代範例角色 ARN。使用您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (*username*)。使用您的帳戶 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. 使用前述請求執行 [CreateApplication](#) 動作以建立應用程式：

```
aws kinesisanalyticstv2 create-application --cli-input-json file://
create_request.json
```

應用程式現在已建立。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 “在 Apache Flink 的受管理服務中設定應用程式登入”](#)。

## 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 kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

## 更新應用程式的程式碼

當您需要使用新版本的程式碼套件更新應用程式程式碼時，請使用[UpdateApplication](#) AWS 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。如需如何使用 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 資料串流”](#)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"
        }
      }
    }
  }
}
```

## 下一步驟

### [步驟 4：清理 AWS 資源](#)



## 步驟 4：清理 AWS 資源

本節包括清理在入門教學課程中建立的 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除路由 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)
- [下一步驟](#)

### 刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

### 刪除您的 Kinesis 資料串流

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。

### 刪除您的 Amazon S3 對象和存儲桶

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。

### 刪除路由 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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 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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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。

### 下一步驟

#### [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：適用於 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。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。
- [AWS 適用於 Amazon MSK 的串流資料解決方案](#)：Amazon MSK 的 AWS 串流資料解決方案提供 AWS CloudFormation 範本，讓資料流經生產者、串流儲存、消費者和目的地。
- [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計程車公司的運營。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。
- [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。

## 開始使用：Flink 1.8.2-棄用

#### Note

阿帕奇 Flink 版本 1.6，1.8 和 1.11 一直沒有得到阿帕奇 Flink 社區的支持超過三年。我們計劃於 2024 年 11 月 5 日在 Amazon 管理服務中淘汰這些版本。從這個日期開始，您將無法為這些 Flink 版本建立新的應用程式。您目前可以繼續執行現有的應用程式。您可以使用適用於 Apache Flink 的 Amazon 受管服務中的就地版本升級功能，以狀態方式升級應用程式。如需詳細資訊，請參閱。[為 Apache Flink 使用就地版本升級](#)

本主題包含使用 Apache Flink 1.8.2 的[教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務教學課程版本](#)。

### 主題

- [適用於 Apache Flink 應用程式的受管理服務元件](#)
- [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：設置 AWS Command Line Interface \( AWS CLI \)](#)
- [第 3 步：為 Apache Flink 應用程序創建並運行託管服務](#)
- [步驟 4：清理 AWS 資源](#)

## 適用於 Apache Flink 應用程式的受管理服務元件

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 執行期生成輸出。

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

- 執行期屬性：您可以使用執行期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[將串流資料來源新增至 Apache Flink 的受管理服務](#)。
- 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[在 Apache Flink 的受管理服務中使用運算子轉換資料](#)。
- 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、Firehose 串流、Amazon S3 儲存貯體等。如需詳細資訊，請參閱[在 Apache Flink 的受管理服務中使用接收器寫入資料](#)。

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[使用阿帕奇 Flink Kinesis 串流連接器與先前的 Apache Flink 版本](#)。
- 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

### 註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

### 若要註冊成為 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。

當您註冊一個時 AWS 帳戶，將創建 AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。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。

### 建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

### 保護您的 AWS 帳戶根使用者

1. 選擇 Root 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 \(主控台\)](#)。

### 建立具有管理權限的使用者

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 身分中心中，將管理存取權授予使用者。

[若要取得有關使用 IAM Identity Center 目錄做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。](#)

以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

如需使用 IAM 身分中心使用者[登入的說明](#)，請參閱[使用AWS 登入者指南中的登入 AWS 存取入口網站](#)。

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「[建立權限集](#)」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「[新增群組](#)」。

授與程式設計存取權

如果使用者想要與 AWS 之外的 AWS Management Console 授與程式設計存取權的方式取決於正在存取的使用者類型。

若要授與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<br>(IAM Identity Center 中管理的使用者) | 使用臨時登入資料來簽署對 AWS CLI、AWS SDK 或 AWS API 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。<br><br>• 如需詳細資訊 AWS CLI，請參閱 <a href="#">《使 AWS CLI 用AWS Command Line Interface者</a> |

| 哪個使用者需要程式設計存取權？ | 到   | By  |
|-----------------|---|---|
|                 |   | <p>指南》AWS IAM Identity Center中的〈配置使用〉。</p> <ul style="list-style-type: none"> <li>如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 <a href="#">IAM 身分中心身分驗證</a>。</li> </ul>   |
| IAM             | 使用臨時登入資料來簽署對 AWS CLI、AWS SDK 或 AWS API 的程式設計要求。         | 遵循《IAM <a href="#">使用者指南</a> 》中的〈 <a href="#">將臨時登入資料搭配 AWS 資源</a> 使用〉中的指示  |
| IAM             | (不建議使用)<br>使用長期認證簽署對 AWS CLI、AWS SDK 或 AWS API 的程式設計要求。 | <p>請依照您要使用的介面所提供的指示操作。</p> <ul style="list-style-type: none"> <li>如需相關資訊 AWS CLI，請參閱使用指南中的<a href="#">使用 IAM 使用者登入資料進行驗證</a>。AWS Command Line Interface</li> <li>對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的<a href="#">使用長期憑據進行身份驗證</a>。</li> <li>如需 AWS API，請參閱 IAM <a href="#">使用者指南</a>中的<a href="#">管理 IAM 使用者的存取金鑰</a>。</li> </ul> |

## 步驟 2：設置 AWS Command Line Interface (AWS CLI)

在此步驟中，您會下載並設定 AWS CLI 要與 Apache Flink 的受管理服務搭配使用。

### 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 AWS CLI 安裝了，則可能需要升級才能獲得最新功能。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](#)。若要檢查的版本 AWS CLI，請執行下列命令：

```
aws --version
```

本自學課程中的練習需要以下 AWS CLI 版本或更高版本：

```
aws-cli/1.16.63
```

**若要設定 AWS CLI**

1. 下載和設定 AWS CLI。如需相關指示，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. 在檔案中新增管理員使用者的具名設定 AWS CLI config 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。若要使用不同的 AWS 區域，請將本自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。

下一步驟

### [第 3 步：為 Apache Flink 應用程式創建並運行託管服務](#)

## 第 3 步：為 Apache Flink 應用程式創建並運行託管服務

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 資料串流](#)
- [將樣本記錄寫入輸入流](#)
- [下載並檢查阿帕奇 Flink 流 Java 代碼](#)
- [編譯應用程式程式碼](#)
- [上傳阿帕奇 Flink 流 Java 代碼](#)
- [建立並執行 Apache Flink 應用程式的受管理服務](#)
- [下一步驟](#)

### 建立兩個 Amazon Kinesis 資料串流

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)，請使用下列 Amazon Kinesis create-stream AWS CLI 指令。

```
$ 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
```

## 將樣本記錄寫入輸入流

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 \(Boto\)](#)。

- 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
```

## 下載並檢查阿帕奇 Flink 流 Java 代碼

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. 導覽至 `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。

如需執行期屬性的詳細資訊，請參閱[在 Apache Flink 的受管理服務中使用執行階段屬性](#)。

## 編譯應用程式程式碼

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) 的相關資訊，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 “使用阿帕奇 Flink Kinesis 串流連接器與先前的 Apache Flink 版本”](#)。

## 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。如果您使用建立應用程式 AWS CLI，請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
```

上傳阿帕奇 Flink 流 Java 代碼

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 Apache Flink 應用程式的受管理服務

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。使用建立應用程式時 AWS CLI，您需要分別建立這些資源。

**主題**

- [創建並運行應用程式 \(控制台\)](#)
- [創建並運行應用程式 \(AWS CLI\)](#)

**創建並運行應用程式 (控制台)**

依照以下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**。
  - 對於執行期，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`

## 編輯身分與存取權管

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 選擇政策。選擇主控台為您在上節所建立的 `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。使用您的帳戶 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                |
|---------------------------------|----------------------------------|------------------|
| <b>ProducerConfigProperties</b> | <b>flink.inputstream.initpos</b> | <b>LATEST</b>    |
| <b>ProducerConfigProperties</b> | <b>aws.region</b>                | <b>us-west-2</b> |
| <b>ProducerConfigProperties</b> | <b>AggregationEnabled</b>        | <b>false</b>     |

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。

## 創建並運行應用程序 ( AWS CLI )

在本節中，您可 AWS CLI 以使用建立和執行 Apache Flink 應用程式的受管理服務。適用於 Apache Flink 的受管理服務會使用指 `kinesisanalyticsv2` AWS CLI 令來建立 Apache Flink 應用程式的受管理服務，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`。使用您的帳戶 ID 取代 Amazon Resource Name (ARN) (`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，選擇角色、建立角色。
3. 在選取可信身分類型下，選擇 AWS 服務。在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)。

選擇下一步：許可。

4. 在連接許可政策頁面，選擇下一步：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，選擇許可標籤。
- b. 選擇連接政策。
- 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 \(主控台\)](#)。

## 為 Apache Flink 應用程式建立受管理的服務

1. 將下列 JSON 程式碼複製到名為 `create_request.json` 的檔案。使用您之前建立之角色的 ARN，取代範例角色 ARN。使用您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (*username*)。使用您的帳戶 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. 使用前述請求執行 [CreateApplication](#) 動作以建立應用程式：

```
aws kinesisanalyticstv2 create-application --cli-input-json file://
create_request.json
```

應用程式現在已建立。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 kinesisanalyticstv2 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 kinesisanalyticstv2 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 “在 Apache Flink 的受管理服務中設定應用程式登入”](#)。

## 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 kinesisanalyticstv2 update-application --cli-input-json file://  
update_properties_request.json
```

## 更新應用程式的程式碼

當您需要使用新版本的程式碼套件更新應用程式程式碼時，請使用[UpdateApplication](#) AWS 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。如需如何使用 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 資料串流”](#)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": "SAMPLEUehYngP87ex1nzYIGYgfhyvDU"  
        }  
      }  
    }  
  }  
}
```



```
}  
}
```

下一步驟

## [步驟 4：清理 AWS 資源](#)

### 步驟 4：清理 AWS 資源

本節包括清理在入門教學課程中建立的 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

#### 刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

#### 刪除您的 Kinesis 資料串流

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。

## 刪除您的 Amazon S3 對象和存儲桶

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。

## 刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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-we st-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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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。

## 開始使用：Flink 1.6.2-棄用

### Note

阿帕奇 Flink 版本 1.6，1.8 和 1.11 一直沒有得到阿帕奇 Flink 社區的支持超過三年。我們計劃於 2024 年 11 月 5 日在 Amazon 管理服務中淘汰這些版本。從這個日期開始，您將無法為這些 Flink 版本建立新的應用程式。您目前可以繼續執行現有的應用程式。您可以使用適用於 Apache Flink 的 Amazon 受管服務中的就地版本升級功能，以狀態方式升級應用程式。如需詳細資訊，請參閱。[為 Apache Flink 使用就地版本升級](#)

本主題包含使用 Apache Flink 1.6.2 的[教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#)教學課程版本。

## 主題

- [適用於 Apache Flink 應用程式的受管理服務元件](#)
- [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：設置 AWS Command Line Interface \( AWS CLI \)](#)
- [第 3 步：為 Apache Flink 應用程序創建並運行託管服務](#)
- [步驟 4：清理 AWS 資源](#)

## 適用於 Apache Flink 應用程式的受管理服務元件

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 執行期生成輸出。

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

- 執行期屬性：您可以使用執行期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[將串流資料來源新增至 Apache Flink 的受管理服務](#)。
- 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[在 Apache Flink 的受管理服務中使用運算子轉換資料](#)。
- 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、Firehose 串流、Amazon S3 儲存貯體等。如需詳細資訊，請參閱[在 Apache Flink 的受管理服務中使用接收器寫入資料](#)。

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

### 註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

### 若要註冊成為 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。

當您註冊一個時 AWS 帳戶，將創建 AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。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。

### 建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

### 保護您的 AWS 帳戶根使用者

1. 選擇 Root 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 \(主控台\)](#)。

### 建立具有管理權限的使用者

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 身分中心中，將管理存取權授予使用者。

[若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。](#)

### 以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

如需使用 IAM 身分中心使用者[登入的說明](#)，請參閱[使用AWS 登入 者指南中的登入 AWS 存取入口網站](#)。

### 指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「[建立權限集](#)」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「[新增群組](#)」。

### 授與程式設計存取權

如果使用者想要與 AWS 之外的 AWS Management Console. 授與程式設計存取 AWS 取權的方式取決於正在存取的使用者類型。

若要授與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<br><br>(IAM Identity Center 中管理的使用者) | 使用臨時登入資料來簽署對 AWS CLI、AWS SDK 或 AWS API 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。 <ul style="list-style-type: none"> <li>• 如需詳細資訊 AWS CLI，請參閱 <a href="#">《使 AWS CLI 用 AWS Command Line Interface 者指南》</a> AWS IAM Identity Center 中的〈配置使用〉。</li> <li>• 如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 <a href="#">IAM 身分中心身分驗證</a>。</li> </ul>  |
| IAM                                       | 使用臨時登入資料來簽署對 AWS CLI、AWS SDK 或 AWS API 的程式設計要求。         | 遵循 <a href="#">《IAM 使用者指南》</a> 中的〈將臨時登入資料搭配 AWS 資源使用〉中的指示  |
| IAM                                       | (不建議使用)<br>使用長期認證簽署對 AWS CLI、AWS SDK 或 AWS API 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。 <ul style="list-style-type: none"> <li>• 如需相關資訊 AWS CLI，請參閱使用指南中的 <a href="#">使用 IAM 使用者登入資料進行驗證</a>。AWS Command Line Interface</li> <li>• 對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的 <a href="#">使用長期憑據進行身份驗證</a>。</li> <li>• 如需 AWS API，請參閱 IAM <a href="#">使用者指南</a> 中的 <a href="#">管理 IAM 使用者的存取金鑰</a>。</li> </ul> |

## 步驟 2：設置 AWS Command Line Interface ( AWS CLI )

在此步驟中，您將下載並設定 AWS CLI 為搭配 Apache Flink 的受管理服務搭配使用。

### 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 AWS CLI 安裝了，則可能需要升級才能獲得最新功能。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](#)。若要檢查的版本 AWS CLI，請執行下列命令：

```
aws --version
```

本自學課程中的練習需要以下 AWS CLI 版本或更高版本：

```
aws-cli/1.16.63
```

### 若要設定 AWS CLI

1. 下載和設定 AWS CLI。如需相關指示，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. 在檔案中新增管理員使用者的具名設定 AWS CLI config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，請參閱中的 [區域和端點 Amazon Web Services 一般參考](#)。

#### 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。

下一步驟

### [第 3 步：為 Apache Flink 應用程式創建並運行託管服務](#)

### 第 3 步：為 Apache Flink 應用程式創建並運行託管服務

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 資料串流](#)
- [將樣本記錄寫入輸入流](#)
- [下載並檢查阿帕奇 Flink 流 Java 代碼](#)
- [編譯應用程式程式碼](#)
- [上傳阿帕奇 Flink 流 Java 代碼](#)
- [建立並執行 Apache Flink 應用程式的受管理服務](#)

#### 建立兩個 Amazon Kinesis 資料串流

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)，請使用下列 Amazon Kinesis create-stream AWS CLI 指令。

```
$ 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
```

### 將樣本記錄寫入輸入流

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
```

## 下載並檢查阿帕奇 Flink 流 Java 代碼

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. 導覽至 `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。

如需執行期屬性的詳細資訊，請參閱[在 Apache Flink 的受管理服務中使用執行階段屬性](#)。

## 編譯應用程式程式碼

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) 的相關資訊，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](#)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。如果您使用建立應用程式 AWS CLI，請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
```

上傳阿帕奇 Flink 流 Java 代碼

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 Apache Flink 應用程式的受管理服務

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。使用建立應用程式時 AWS CLI，您需要分別建立這些資源。

主題

- [創建並運行應用程序 \(控制台\)](#)
- [創建並運行應用程序 \(AWS CLI\)](#)

創建並運行應用程序 (控制台)

依照以下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**。
  - 對於執行期，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。

- 對於存取許可，選擇建立/更新 IAM 角色 **kinesis-analytics-MyApplication-us-west-2**。
- 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`

## 編輯身分與存取權管

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

- 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
- 選擇政策。選擇主控台為您在上節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
- 在摘要頁面，選擇編輯政策。請選擇 JSON 標籤。
- 將下列政策範例的反白部分新增至政策。使用您的帳戶 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                |
|---------------------------------|----------------------------------|------------------|
| <b>ProducerConfigProperties</b> | <b>flink.inputstream.initpos</b> | <b>LATEST</b>    |
| <b>ProducerConfigProperties</b> | <b>aws.region</b>                | <b>us-west-2</b> |
| <b>ProducerConfigProperties</b> | <b>AggregationEnabled</b>        | <b>false</b>     |

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。

## 創建並運行應用程序 ( AWS CLI )

在本節中，您可 AWS CLI 以使用建立和執行 Apache Flink 應用程式的受管理服務。適用於 Apache Flink 的受管理服務會使用指kinesisanalyticsv2 AWS CLI 令來建立 Apache Flink 應用程式的受管理服務，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*。使用您的帳戶 ID 取代 Amazon Resource Name (ARN) (*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，選擇角色、建立角色。
3. 在選取可信身分類型下，選擇 AWS 服務。在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)。

選擇下一步：許可。

4. 在連接許可政策頁面，選擇下一步：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，選擇許可標籤。
- b. 選擇連接政策。
- 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 \(主控台\)](#)。

## 為 Apache Flink 應用程式建立受管理的服務

1. 將下列 JSON 程式碼複製到名為 `create_request.json` 的檔案。使用您之前建立之角色的 ARN，取代範例角色 ARN。使用您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (*username*)。使用您的帳戶 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. 使用前述請求執行 [CreateApplication](#) 動作以建立應用程式：

```
aws kinesisanalyticstv2 create-application --cli-input-json file://
create_request.json
```

應用程式現在已建立。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 kinesisanalyticstv2 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 kinesisanalyticstv2 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 “在 Apache Flink 的受管理服務中設定應用程式登入”](#)。

## 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 kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

## 更新應用程式的程式碼

當您需要使用新版本的程式碼套件更新應用程式程式碼時，請使用[UpdateApplication](#) AWS 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 資料串流”](#)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：清理 AWS 資源

本節包括清理在入門教學課程中建立的 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

刪除您的 Kinesis 資料串流

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。

刪除您的 Amazon S3 對象和存儲桶

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。



## 刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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 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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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。

## 適用於 Apache Flink 的受管理服務的舊版 (舊版) 範例

### Note

如需目前範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

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](#)
- [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#)

**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](#)
- [斯卡拉例子](#)

## 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：翻滾窗](#)
- [範例：滑動窗](#)
- [範例：寫入 Amazon S3 儲存貯體](#)
- [教學課程：使用 Apache Flink 應用程式的受管理服務，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：搭配 Kinesis 資料串流使用 EFO 取用者](#)
- [例子：寫信給 Firehose](#)
- [範例：從不同帳戶的 Kinesis 串流讀取](#)
- [教學課程：搭配 Amazon MSK 使用自訂信任庫](#)

## 範例：翻滾窗

**Note**

如需目前範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

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 [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#) 練習。

本主題包含下列章節：

- [建立相依資源](#)
- [將樣本記錄寫入輸入流](#)
- [下載並檢查應用程式程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳阿帕奇 Flink 流 Java 代碼](#)
- [建立並執行 Apache Flink 應用程式的受管理服務](#)
- [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。如需建立這些資源的相關指示，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。

## 將樣本記錄寫入輸入流

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. 導覽至 `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：

```
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，請先安裝。如需詳細資訊，請參閱[教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#)教學課程中的[完成必要的先決條件](#)。
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)。

### 上傳阿帕奇 Flink 流 Java 代碼

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 Apache Flink 應用程式的受管理服務

依照以下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**。
  - 對於執行期，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`

## 編輯身分與存取權管

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 選擇政策。選擇主控台為您在上節所建立的 **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。使用您的帳戶 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. 若要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「暫停視窗」教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

### 刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

### 刪除您的 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，選擇「ExampleOutputStream 動作」，選擇「刪除」，然後確認刪除。

### 刪除您的 Amazon S3 對象和存儲桶

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。

## 刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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-we st-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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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：滑動窗

### Note

如需目前範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

### Note

若要設定此練習的必要先決條件，請先完成 [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#) 練習。

本主題包含下列章節：

- [建立相依資源](#)
- [將樣本記錄寫入輸入流](#)

- [下載並檢查應用程式程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳阿帕奇 Flink 流 Java 代碼](#)
- [建立並執行 Apache Flink 應用程式的受管理服務](#)
- [清理 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。如需建立這些資源的相關指示，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。

## 將樣本記錄寫入輸入流

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. 導覽至 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));
```

- 該應用程式使用 `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：

```
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，請先安裝。如需詳細資訊，請參閱[教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#)教學課程中的[完成必要的先決條件](#)。
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)。

### 上傳阿帕奇 Flink 流 Java 代碼

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 Apache Flink 應用程式的受管理服務

依照以下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**。
  - 對於執行期，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`

**編輯身分與存取權管**

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 選擇政策。選擇主控台為您在上節所建立的 **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。使用您的帳戶 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. 若要 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原則，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。

## 清理 AWS 資源

本節包括清理在「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：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

### 刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

### 刪除您的 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，選擇「ExampleOutputStream 動作」，選擇「刪除」，然後確認刪除。

### 刪除您的 Amazon S3 對象和存儲桶

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。

#### 刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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 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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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 [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#) 練習。

本主題包含下列章節：

- [建立相依資源](#)
- [將樣本記錄寫入輸入流](#)

- [下載並檢查應用程式程式碼](#)
- [修改應用程式程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳阿帕奇 Flink 流 Java 代碼](#)
- [建立並執行 Apache Flink 應用程式的受管理服務](#)
- [驗證應用程式輸出](#)
- [可選：自訂來源和接收器](#)
- [清理 AWS 資源](#)

## 建立相依資源

在為本練習建立 Managed Service for Apache Flink 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和輸出 (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。如需建立這些資源的相關指示，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。

## 將樣本記錄寫入輸入流

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. 導覽至 `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，請先安裝。如需詳細資訊，請參閱 [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#) 教學課程中的 [完成必要的先決條件](#)。
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。

## 上傳阿帕奇 Flink 流 Java 代碼

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 Apache Flink 應用程式的受管理服務

依照以下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**。
  - 對於執行期，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**。
- 對於執行期，選擇 Apache Flink。
- 將版本保留為 Apache Flink 1.15.2 版 (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`

### 編輯身分與存取權管

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 選擇政策。選擇主控台為您在上節所建立的 **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。使用您的帳戶 ID 取代範例帳戶 ID (`012345678901`)。使用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. 若要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'
```

可選：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 “上傳阿帕奇 Flink 流 Java 代碼”](#) 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，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](#)。

## 清理 AWS 資源

本節包括清理您在 Amazon S3 教學中建立的 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

## 刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。



## 刪除您的 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。

## 刪除您的 Amazon S3 對象和存儲桶

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。

## 刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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-we st-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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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 Apache Flink 應用程式的受管理服務，將 MSK 叢集中某個主題的資料複製到 VPC 中的另一個主題

**Note**

如需目前範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

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 [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#) 練習。

本教學課程包含下列章節：

- [建立 Amazon VPC 和 Amazon MSK 叢集](#)
- [建立應用程式程式碼](#)
- [上傳阿帕奇 Flink 流 Java 代碼](#)
- [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](#) 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 AWS KafkaTutorialTopicDestination
```

- 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.aws-kafka-tutorial-cluster.t79r6y.c4.kafka.us-west-2.amazonaws.com:9094,b-1.aws-kafka-tutorial-cluster.t79r6y.c4.kafka.us-west-2.amazonaws.com:9094,b-3.aws-kafka-tutorial-cluster.t79r6y.c4.kafka.us-west-2.amazonaws.com:9094"
}
```

- 按照自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，

## 上傳阿帕奇 Flink 流 Java 代碼

在本節中，您會將應用程式的程式碼上傳至在[教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#)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 “上傳應用程式代碼JAR文件”](#) 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. 在 <https://console.aws.amazon.com/flink> 開啟適用於阿帕奇 Flink 主控台的受管理服務。
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**。
  - 針對執行期，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   |
|--------------------|-------------------------|---|
| <b>KafkaSource</b> | 主題                      | AWS KafkaTutorialTopic                                    |
| <b>KafkaSource</b> | bootstrap.servers       | <i>#####</i>  |
| <b>KafkaSource</b> | security.protocol       | SSL   |
| <b>KafkaSource</b> | ssl.truststore.location | /usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts |

| 群組 ID              | 金鑰                      | 值        |
|--------------------|-------------------------|----------|
| <b>KafkaSource</b> | 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   |
|------------------|-------------------------|---|
| <b>KafkaSink</b> | 主題                      | AWS KafkaTutorialTopic Destination                        |
| <b>KafkaSink</b> | bootstrap.servers       | #####   |
| <b>KafkaSink</b> | security.protocol       | SSL   |
| <b>KafkaSink</b> | ssl.truststore.location | /usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts |
| <b>KafkaSink</b> | ssl.truststore.password | changeit  |
| <b>KafkaSink</b> | 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。如需如何使用屬性的詳細資訊，請參閱[在 Apache Flink 的受管理服務中使用執行階段屬性](#)。

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。

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 AWS KafkaTutorialTopicDestination --from-  
beginning
```

如果目的地主題中未顯示任何記錄，請參閱 [疑難排解 Apache Flink 的受管理服務](#) 主題中的 [無法訪問中的資源 VPC](#) 一節。

範例：搭配 Kinesis 資料串流使用 EFO 取用者

### Note

如需目前範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

在本練習中，您會建立 Apache Flink 應用程式的受管服務，該應用程式會使用[增強型散發 \(EFO\)](#) 用戶從 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 [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#) 練習。

本主題包含下列章節：

- [建立相依資源](#)
- [將樣本記錄寫入輸入流](#)
- [下載並檢查應用程式程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳阿帕奇 Flink 流 Java 代碼](#)
- [建立並執行 Apache Flink 應用程式的受管理服務](#)
- [清理 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。如需建立這些資源的相關指示，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。

將樣本記錄寫入輸入流

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. 導覽至 `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，請先安裝。如需詳細資訊，請參閱[教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#)教學課程中的[完成必要的先決條件](#)。
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)。

### 上傳阿帕奇 Flink 流 Java 代碼

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 Apache Flink 應用程式的受管理服務

依照以下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**。
  - 對於執行期，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`

## 編輯身分與存取權管

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 選擇政策。選擇主控台為您在上節所建立的 **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。使用您的帳戶 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.recorderpublisher | EF0                 |
| ConsumerConfigProperties | flink.stream.efs.consumername  | basic-efs-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           |

| 群組 ID                           | 金鑰                        | 值            |
|---------------------------------|---------------------------|--------------|
| <b>ProducerConfigProperties</b> | <b>AggregationEnabled</b> | <b>false</b> |

8. 在監控下，確保監控指標層級設為應用程式。
9. 若要 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。

您也可以從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`)。

## 清理 AWS 資源

本節包括清理在 efo 視窗教學課程中建立的 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)



- [刪除 Amazon S3 物件與儲存貯體](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

### 刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

### 刪除您的 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，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。

### 刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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-we st-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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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。

例子：寫信給 Firehose

**Note**

如需目前範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

在本練習中，您會建立 Apache Flink 應用程式的受管理服務，該應用程式具有 Kinesis 資料串流做為來源，並將 Firehose 串流做為接收器。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 [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#) 練習。

本節包含下列步驟：

- [建立相依資源](#)
- [將樣本記錄寫入輸入流](#)
- [下載並檢查阿帕奇 Flink 流 Java 代碼](#)
- [編譯應用程式程式碼](#)
- [上傳阿帕奇 Flink 流 Java 代碼](#)
- [建立並執行 Apache Flink 應用程式的受管理服務](#)
- [清除 AWS 資源](#)

## 建立相依資源

在為本練習建立 Managed Service for Apache Flink 之前，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 Firehose 資料流。
- Amazon S3 儲存貯體，用來儲存應用程式的程式碼 (ka-app-code-*<username>*)

您可以使用主控台建立 Kinesis 串流、Amazon S3 儲存貯體和 Firehose 串流。如需建立這些資源的相關指示，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。
- 在[亞馬遜資料防火管開發人員指南中建立 Amazon Kinesis 資料 Firehose 交付串流](#)。為您的「Firehose」串流 **ExampleDeliveryStream** 命名。當您建立 Firehose 串流時，請同時建立 Firehose 串流的 S3 目標和 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。

## 將樣本記錄寫入輸入流

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。

## 下載並檢查阿帕奇 Flink 流 Java 代碼

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. 導覽至 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));
```

- 該應用程式使用 Firehose 接收器將數據寫入到 Firehose 流。下列程式碼片段會建立 Firehose 收槽：

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

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

## 編譯應用程式程式碼

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，請先安裝。如需詳細資訊，請參閱[教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#)教學課程中的[完成必要的先決條件](#)。
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 “使用阿帕奇 Flink Kinesis 串流連接器與先前的 Apache Flink 版本”](#)。
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)。

## 上傳阿帕奇 Flink 流 Java 代碼

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 Apache Flink 應用程式的受管理服務

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。使用建立應用程式時 AWS CLI，您需要分別建立這些資源。

### 主題

- [創建並運行應用程序 \(控制台\)](#)
- [創建並運行應用程序 \(AWS CLI\)](#)


### 創建並運行應用程序 (控制台)

依照以下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**。
  - 對於執行期，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**

## 編輯身分與存取權管

編輯身分與存取權管理政策，以新增存取 Kinesis 資料串流和 Firehose 串流的許可。

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 選擇政策。選擇主控台為您在上節所建立的 **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。使用您的帳戶 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. 若要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。

## 創建並運行應用程序 ( AWS CLI )

在本節中，您可 AWS CLI 以使用建立和執行 Apache Flink 應用程式的受管理服務。

### 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。使用您用於建立 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*。使用您的帳戶 ID 取代 Amazon Resource Name (ARN) (*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，選擇角色、建立角色。
3. 在選取可信身分類型下，選擇 AWS 服務。在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)。

選擇下一步：許可。

4. 在連接許可政策頁面，選擇下一步：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，選擇許可標籤。
- b. 選擇連接政策。
- 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 \(主控台\)](#)。

## 為 Apache Flink 應用程式建立受管理的服務

1. 將下列 JSON 程式碼複製到名為 `create_request.json` 的檔案。使用您之前建立之角色的 ARN，取代範例角色 ARN。使用您在[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。使用您的帳戶 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. 使用前述請求執行 [CreateApplication](#) 動作以建立應用程式：

```
aws kinesisanalyticstv2 create-application --cli-input-json file://
create_request.json
```

應用程式現在已建立。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 “在 Apache Flink 的受管理服務中設定應用程式登入”](#)。

## 更新應用程式的程式碼

當您需要使用新版本的程式碼套件更新應用程式程式碼時，請使用[UpdateApplication](#) AWS 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

本節包括清理在入門教學課程中建立的 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)

- [刪除您的「Firehose」串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

刪除您的 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。

刪除您的「Firehose」串流

1. 在以下網址開啟 Kinesis 主控台：<https://console.aws.amazon.com/kinesis>。
2. 在「Firehose」面板中，選擇 ExampleDeliveryStream。
3. 在 ExampleDeliveryStream 頁面中，選擇「刪除 Firehose 串流」，然後確認刪除。

刪除您的 Amazon S3 對象和存儲桶

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 Firehose 串流的目的地建立了 Amazon S3 儲存貯體，請也刪除該儲存貯體。



## 刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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 原則。
5. 選擇政策動作，然後選擇刪除。
6. 如果您已為 Firehose 串流建立新政策，請同時刪除該原則。
7. 在導覽列中，選擇角色。
8. 選擇運動分析-MyApplication us-west-2 角色。
9. 選擇刪除角色，然後確認刪除。
10. 如果您為 Firehose 串流建立了新角色，請同時刪除該角色。

## 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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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 串流讀取

### Note

如需目前範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

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](#)

- [建立源 Kinesis 串流](#)
- [建立和更新 IAM 角色和政策](#)
- [更新 Python 腳本](#)
- [更新 Java 應用程式](#)
- [建置、上傳及執行應用程式](#)

## 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[教學課程：開始使用適用於 Apache Flink 的 DataStream API受管理服務](#)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。使用您 AWS 用於應用程式和接收串流入門教學課程的帳戶。對來源串流使用不同的 AWS 帳戶。

## 設定

您將使用命名的配置文件訪問您的兩個 AWS 帳戶。修改您的 AWS 認證和設定檔，以包含兩個設定檔，其中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`。它作為如何使用配置文件訪問兩個不同 AWS 帳戶的示例。

若要取得有關搭配使用具名設定檔的詳細資訊 AWS CLI，請參閱文件中的 [具名設定AWS Command Line Interface](#) 檔。

## 建立源 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 帳戶存取物件，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。如需建立和管理 IAM 角色和政策的相關資訊，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。使用將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
```

## 來源帳號角色和策略

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。使用來源帳戶 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 AWS 服務。在服務清單中，選擇 Kinesis。
  3. 在選取使用案例區塊中，選擇 Managed Service for Apache Flink。
  4. 選擇下一步：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。使用您的接收器帳戶 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 腳本

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。使用您的來源帳戶 ID 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、上傳及執行應用程式

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，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 使用自訂信任庫

#### Note

如需目前範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

## 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 Amazon MSK Config 提供者公用程式。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 [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#) 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。

為了通過 TLS 安全地連接任何卡夫卡客戶端到卡夫卡集群，卡夫卡客戶端 (如示例 Flink 應用程式) 必須信任卡夫卡集群的服務器證書 (從發行 CA 到根級 CA) 提供的完整信任鏈。作為自訂信任存放區的範例，我們將使用已啟用相互 TLS (MTLS) 身份驗證的 Amazon MSK 叢集。這表示 MSK 叢集節點使用由 Certificate Manager 專用憑 AWS 證授權單位 (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**

- 密鑰庫用於存儲應用程式應該向服務器或客戶端進行驗證的私鑰和身份證書。
- 信任庫用於存儲來自認證授權單位 (CA) 的證書，以驗證服務器在 SSL 連接中提供的證書。

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 \(Amazon EC2 或亞馬遜 EKS\) 的自定義阿帕奇卡夫卡集群](#)
- [一致性的卡夫卡](#) 集群託管在 AWS
- 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：

- [使用 Amazon MSK 叢集建立 VPC](#)
- [建立自訂信任庫並將其套用至叢集](#)
- [建立應用程式程式碼](#)
- [上傳阿帕奇 Flink 流 Java 代碼](#)
- [建立應用程式](#)
- [設定應用程式](#)
- [執行應用程式](#)
- [測試應用程式。](#)

## 使用 Amazon MSK 叢集建立 VPC

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](#) 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 地址的所有流量。

- 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條件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 Region。

### 建立自訂信任庫並將其套用至叢集

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，請按照 Amazon Managed Streaming for Apache Kafka 開發人員指南中的[用戶端身分驗證](#)教學課程操作。

### 建立應用程式程式碼

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。

## 上傳阿帕奇 Flink 流 Java 代碼

在本節中，您會將應用程式的程式碼上傳至在[教學課程：開始使用適用於 Apache Flink 的 DataStream API受管理服務](#)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 “上傳應用程式代碼JAR文件”](#) 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**。
  - 針對執行期，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  |
|--------------------------|--------------------------------------|--|
| <code>KafkaSource</code> | 主題                                   | AWS KafkaTutorialTopic   |
| <code>KafkaSource</code> | <code>bootstrap.servers</code>       | <code>#####</code>   |
| <code>KafkaSource</code> | <code>security.protocol</code>       | SSL  |
| <code>KafkaSource</code> | <code>ssl.truststore.location</code> | <code>/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts</code> |
| <code>KafkaSource</code> | <code>ssl.truststore.password</code> | 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   |
|------------------|-------------------------|---|
| <b>KafkaSink</b> | 主題                      | AWS KafkaTutorialTopic Destination                        |
| <b>KafkaSink</b> | bootstrap.servers       | #####   |
| <b>KafkaSink</b> | security.protocol       | SSL   |
| <b>KafkaSink</b> | ssl.truststore.location | /usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts |
| <b>KafkaSink</b> | ssl.truststore.password | changeit  |
| <b>KafkaSink</b> | 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。如需如何使用屬性的詳細資訊，請參閱[在 Apache Flink 的受管理服務中使用執行階段屬性](#)。

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。

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 AWS KafkaTutorialTopicDestination --from-  
beginning
```

如果目的地主題中未顯示任何記錄，請參閱 [疑難排解 Apache Flink 的受管理服務](#) 主題中的 [無法訪問中的資源 VPC](#) 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

- [示例：在 Python 中創建一個翻滾窗口](#)
- [示例：在 Python 中創建一個滑動窗口](#)
- [範例：使用 Python 將串流資料傳送至 Amazon S3](#)

示例：在 Python 中創建一個翻滾窗口

### Note

如需目前範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

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 [教學課程：開始在 Apache Flink 的受管理服務中使用 Python](#) 練習。

本主題包含下列章節：

- [建立相依資源](#)
- [將樣本記錄寫入輸入流](#)
- [下載並檢查應用程式程式碼](#)
- [壓縮並上傳阿帕奇 Flink 流 Python 代碼](#)
- [建立並執行 Apache Flink 應用程式的受管理服務](#)
- [清理 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。如需建立這些資源的相關指示，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。

## 將樣本記錄寫入輸入流

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 AWS CLI 將您的配置為使用您的帳戶憑據和默認區域。若要設定您的 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. 導覽至 `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")

```

- 應用程式使用來自 [flink-sql-connector-kinesis-1.15.2.jar](#) 的 Kinesis Flink 連接器。

## 壓縮並上傳阿帕奇 Flink 流 Python 代碼

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 Apache Flink 應用程式的受管理服務

依照以下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**。
  - 對於執行期，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                               |
|--------------------------------|----------------------------------|---------------------------------|
| <code>consumer.config.0</code> | <code>input.stream.name</code>   | <code>ExampleInputStream</code> |
| <code>consumer.config.0</code> | <code>aws.region</code>          | <code>us-west-2</code>          |
| <code>consumer.config.0</code> | <code>scan.stream.initpos</code> | <code>LATEST</code>             |

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                                |
|--------------------------------|---------------------------------|----------------------------------|
| <code>producer.config.0</code> | <code>output.stream.name</code> | <code>ExampleOutputStream</code> |
| <code>producer.config.0</code> | <code>aws.region</code>         | <code>us-west-2</code>           |
| <code>producer.config.0</code> | <code>shard.count</code>        | <code>1</code>                   |

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 [指定您的程式碼檔](#)。
9. 輸入下列資料：

| 群組 ID  | 金鑰                   | 值   |
|--|----------------------|---|
| <code>kinesis.analytics.flink.run.options</code> | <code>python</code>  | <code>tumbling-windows.py</code>                    |
| <code>kinesis.analytics.flink.run.options</code> | <code>jarfile</code> | <code>flink-sql-connector-kinesis-1.15.2.jar</code> |



10. 在監控下，確保監控指標層級設為應用程式。
11. 若要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。

## 編輯身分與存取權管

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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。使用您的帳戶 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。

## 清理 AWS 資源

本節包括清除在「暫停視窗」教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

### 刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

### 刪除您的 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，選擇「ExampleOutputStream 動作」，選擇「刪除」，然後確認刪除。

### 刪除您的 Amazon S3 對象和存儲桶

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。

#### 刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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 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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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。

示例：在 Python 中創建一個滑動窗口

#### Note

如需目前範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

#### Note

若要設定此練習的必要先決條件，請先完成 [教學課程：開始在 Apache Flink 的受管理服務中使用 Python 練習](#)。

本主題包含下列章節：

- [建立相依資源](#)
- [將樣本記錄寫入輸入流](#)
- [下載並檢查應用程式程式碼](#)
- [壓縮並上傳阿帕奇 Flink 流 Python 代碼](#)
- [建立並執行 Apache Flink 應用程式的受管理服務](#)
- [清理 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。如需建立這些資源的相關指示，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。

## 將樣本記錄寫入輸入流

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 AWS CLI 將您的配置為使用您的帳戶憑據和默認區域。若要設定您的 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. 導覽至 `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")  
    )
```

- 應用程式使用 Kinesis Flink 連接器，從 [flink-sql-connector-kinesis- 1.15.2.jar](#) 檔案。

## 壓縮並上傳阿帕奇 Flink 流 Python 代碼

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 Apache Flink 應用程式的受管理服務

依照以下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**。
  - 對於執行期，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                               |
|--------------------------------|----------------------------------|---------------------------------|
| <code>consumer.config.0</code> | <code>input.stream.name</code>   | <code>ExampleInputStream</code> |
| <code>consumer.config.0</code> | <code>aws.region</code>          | <code>us-west-2</code>          |
| <code>consumer.config.0</code> | <code>scan.stream.initpos</code> | <code>LATEST</code>             |

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                                |
|--------------------------------|---------------------------------|----------------------------------|
| <code>producer.config.0</code> | <code>output.stream.name</code> | <code>ExampleOutputStream</code> |
| <code>producer.config.0</code> | <code>aws.region</code>         | <code>us-west-2</code>           |
| <code>producer.config.0</code> | <code>shard.count</code>        | <code>1</code>                   |

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 [指定您的程式碼檔](#)。
9. 輸入以下應用程式屬性和數值：

| 群組 ID  | 金鑰                   | 值   |
|--|----------------------|---|
| <code>kinesis.analytics.flink.run.options</code> | <code>python</code>  | <code>sliding-windows.py</code>                     |
| <code>kinesis.analytics.flink.run.options</code> | <code>jarfile</code> | <code>flink-sql-connector-kinesis_1.15.2.jar</code> |

10. 在監控下，確保監控指標層級設為應用程式。
11. 若要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。

## 編輯身分與存取權管

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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。使用您的帳戶 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。

## 清理 AWS 資源

本節包括清理在「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：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

### 刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

### 刪除您的 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，選擇「ExampleOutputStream 動作」，選擇「刪除」，然後確認刪除。

### 刪除您的 Amazon S3 對象和存儲桶

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。

#### 刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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-we st-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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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

#### Note

如需目前範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

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 [教學課程：開始在 Apache Flink 的受管理服務中使用 Python 練習](#)。

本主題包含下列章節：

- [建立相依資源](#)
- [將樣本記錄寫入輸入流](#)
- [下載並檢查應用程式程式碼](#)
- [壓縮並上傳阿帕奇 Flink 流 Python 代碼](#)
- [建立並執行 Apache Flink 應用程式的受管理服務](#)
- [清理 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和輸出 (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。如需建立這些資源的相關指示，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。

## 將樣本記錄寫入輸入流

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 AWS CLI 將您的配置為使用您的帳戶憑據和默認區域。若要設定您的 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. 導覽至 `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)

```

- 應用程式使用 Kinesis Flink 連接器，從 [flink-sql-connector-kinesis-1.15.2.jar](#) 檔案。

## 壓縮並上傳阿帕奇 Flink 流 Python 代碼

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 Apache Flink 應用程式的受管理服務

依照以下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**。
  - 對於執行期，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                               |
|--------------------------------|----------------------------------|---------------------------------|
| <code>consumer.config.0</code> | <code>input.stream.name</code>   | <code>ExampleInputStream</code> |
| <code>consumer.config.0</code> | <code>aws.region</code>          | <code>us-west-2</code>          |
| <code>consumer.config.0</code> | <code>scan.stream.initpos</code> | <code>LATEST</code>             |

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 [指定您的程式碼檔](#)。
7. 輸入以下應用程式屬性和數值：

| 群組 ID  | 金鑰                   | 值  |
|--|----------------------|--|
| <code>kinesis.analytics.flink.run.options</code> | <code>python</code>  | <code>streaming-file-sink.py</code>                            |
| <code>kinesis.analytics.flink.run.options</code> | <code>jarfile</code> | <code>S3Sink/lib/flink-sql-connector-kinesis-1.15.2.jar</code> |

- 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 [指定您的程式碼檔](#)。
- 輸入以下應用程式屬性和值：(使用 Amazon S3 儲存貯體的實際名稱取代 `bucket-name`。)

| 群組 ID                      | 金鑰                              | 值                               |
|----------------------------|---------------------------------|---------------------------------|
| <code>sink.config.0</code> | <code>output.bucket.name</code> | <code><i>bucket-name</i></code> |

- 在監控下，確保監控指標層級設為應用程式。
- 若要 CloudWatch 記錄，請選取 [啟用] 核取方塊。
- 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。

## 編輯身分與存取權管

編輯 IAM 政策來新增存取 Kinesis 資料串流的許可。

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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。使用您的帳戶 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。

## 清理 AWS 資源

本節包括清理在「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：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

### 刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

### 刪除您的 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。

### 刪除您的 Amazon S3 對象和存儲桶

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。

### 刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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-we st-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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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。

### 斯卡拉例子

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

- [示例：在 Scala 中創建一個翻滾窗口](#)
- [示例：在 Scala 中創建一個滑動窗口](#)
- [示例：將流數據發送到斯卡拉中的 Amazon S3](#)

示例：在 Scala 中創建一個翻滾窗口

#### Note

如需目前範例，請參閱[為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

#### 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 相依性添加到他們的 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式，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程式碼](#)
- [編譯和上傳應用程式的程式碼](#)
- [創建並運行應用程式 \(控制台\)](#)
- [建立並執行應用程式 \(CLI\)](#)
- [更新應用程式的程式碼](#)
- [清理 AWS 資源](#)

下載並檢查應用程式程式碼

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. 導覽至 `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, 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。會讀取執行期應用程式的屬性，來設定連接器。如需執行期屬性的詳細資訊，請參閱[執行期屬性](#)。

## 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。

## 創建並運行應用程序 (控制台)

依照以下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**。
  - 對於執行期，選擇 Apache Flink。
  - 將版本保留為 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

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                         |
|---------------------------------|-----------------------------|---------------------------|
| <b>ConsumerConfigProperties</b> | <b>input.stream.name</b>    | <b>ExampleInputStream</b> |
| <b>ConsumerConfigProperties</b> | <b>aws.region</b>           | <b>us-west-2</b>          |
| <b>ConsumerConfigProperties</b> | <b>flink.stream.initpos</b> | <b>LATEST</b>             |

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                          |
|---------------------------------|---------------------------|----------------------------|
| <b>ProducerConfigProperties</b> | <b>output.stream.name</b> | <b>ExampleOutputStream</b> |
| <b>ProducerConfigProperties</b> | <b>aws.region</b>         | <b>us-west-2</b>           |

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

**編輯身分與存取權管**

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 S3 儲存貯體許可

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 選擇政策。選擇主控台為您在上節所建立的 **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。使用您的帳戶 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 AWS Command Line Interface 以使用建立和執行 Apache Flink 應用程式的受管理服務。使用指 AWS CLI 令為 Apache Flink 應用程式建立受管理的服務，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**。使用您的帳戶 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. 前往 <https://console.aws.amazon.com/iam/> 開啟 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. 選擇下一步：許可。
7. 在連接許可政策頁面，選擇下一步：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步驟 [建立許可政策](#) 中建立的政策。

- a. 在摘要頁面，選擇許可標籤。
- b. 選擇連接政策。
- 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。使用您之前建立之角色的 ARN，取代範例角色 ARN。使用您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (username)。使用您的帳

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式：

```
aws kinesisanalyticstv2 create-application --cli-input-json file://create_request.json
```

應用程式現在已建立。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，您將使用 [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": "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 kinesisanalyticstv2 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

在本節中，您可以使用 [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": "tumbling_window"
}
```

2. 使用前述請求執行 `StopApplication` 動作以停止應用程式：

```
aws kinesisanalyticstv2 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，而無需重新編譯應用程式程式碼。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 kinesisanalyticstv2 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 [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。如需如何使用 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"
        }
      }
    }
  }
}

```

```
        }  
    }  
}  
}
```

## 清理 AWS 資源

本節包括清理在「暫停視窗」教學課程中建立的 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

### 刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

### 刪除您的 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，選擇「ExampleOutputStream 動作」，選擇「刪除」，然後確認刪除。

### 刪除您的 Amazon S3 對象和存儲桶

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。

#### 刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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-we st-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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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。

示例：在 Scala 中創建一個滑動窗口

#### Note

如需目前範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

#### 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 相依性添加到他們的 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式，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程式碼](#)
- [編譯和上傳應用程式的程式碼](#)
- [創建並運行應用程式 \(控制台\)](#)
- [建立並執行應用程式 \(CLI\)](#)
- [更新應用程式的程式碼](#)
- [清理 AWS 資源](#)

### 下載並檢查應用程式程式碼

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. 導覽至 `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
}
```

- 該應用程式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。會讀取執行期應用程式的屬性，來設定連接器。如需執行期屬性的詳細資訊，請參閱[執行期屬性](#)。

## 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。

## 創建並運行應用程序 (控制台)

依照以下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**。
  - 對於執行期，選擇 Apache Flink。
  - 將版本保留為 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

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                         |
|---------------------------------|-----------------------------|---------------------------|
| <b>ConsumerConfigProperties</b> | <b>input.stream.name</b>    | <b>ExampleInputStream</b> |
| <b>ConsumerConfigProperties</b> | <b>aws.region</b>           | <b>us-west-2</b>          |
| <b>ConsumerConfigProperties</b> | <b>flink.stream.initpos</b> | <b>LATEST</b>             |

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                          |
|---------------------------------|---------------------------|----------------------------|
| <b>ProducerConfigProperties</b> | <b>output.stream.name</b> | <b>ExampleOutputStream</b> |
| <b>ProducerConfigProperties</b> | <b>aws.region</b>         | <b>us-west-2</b>           |

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

**編輯身分與存取權管**

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 S3 儲存貯體許可

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 選擇政策。選擇主控台為您在上節所建立的 **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。使用您的帳戶 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 AWS Command Line Interface 以使用建立和執行 Apache Flink 應用程式的受管理服務。使用指 AWS CLI 令為 Apache Flink 應用程式建立受管理的服務，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**。使用您的帳戶 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. 前往 <https://console.aws.amazon.com/iam/> 開啟 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. 選擇下一步：許可。
7. 在連接許可政策頁面，選擇下一步：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，選擇許可標籤。
- b. 選擇連接政策。
- 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。使用您之前建立之角色的 ARN，取代範例角色 ARN。使用您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (username)。使用您的帳戶 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式：

```
aws kinesisanalyticstv2 create-application --cli-input-json file://create_request.json
```

應用程式現在已建立。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，您將使用 [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": "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 kinesisanalyticstv2 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

在本節中，您可以使用 [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": "sliding_window"
}
```

## 2. 使用前述請求執行 StopApplication 動作以停止應用程式：

```
aws kinesisanalyticstv2 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，而無需重新編譯應用程式程式碼。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 kinesisanalyticstv2 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 [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。如需如何使用 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": "SAMPLEUehYngP87ex1nzYIGYgfhyvDU"
        }
      }
    }
  }
}
```

```
}  
}  
}
```

## 清理 AWS 資源

本節包括清理滑動式視窗教學課程中建立的 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

### 刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

### 刪除您的 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，選擇「ExampleOutputStream 動作」，選擇「刪除」，然後確認刪除。

### 刪除您的 Amazon S3 對象和存儲桶

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。



## 刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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-we st-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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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。

示例：將流數據發送到斯卡拉中的 Amazon S3

### Note

如需目前範例，請參閱 [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)。

### 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 相依性添加到他們的 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程式碼](#)
- [編譯和上傳應用程式的程式碼](#)
- [創建並運行應用程序 \(控制台\)](#)
- [建立並執行應用程式 \(CLI\)](#)
- [更新應用程式的程式碼](#)
- [清理 AWS 資源](#)

### 下載並檢查應用程式程式碼

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. 導覽至 `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)
}
```

該應用程式還使 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。會讀取執行期應用程式的屬性，來設定連接器。如需執行期屬性的詳細資訊，請參閱[執行期屬性](#)。

## 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。

### 創建並運行應用程序 (控制台)

依照以下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**。
  - 對於執行期，選擇 Apache Flink。
  - 將版本保留為 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

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                               |
|---------------------------------------|-----------------------------------|---------------------------------|
| <code>ConsumerConfigProperties</code> | <code>input.stream.name</code>    | <code>ExampleInputStream</code> |
| <code>ConsumerConfigProperties</code> | <code>aws.region</code>           | <code>us-west-2</code>          |
| <code>ConsumerConfigProperties</code> | <code>flink.stream.initpos</code> | <code>LATEST</code>             |

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   |
|---------------------------------|---------------------|---|
| <b>ProducerConfigProperties</b> | <b>s3.sink.path</b> | <b>s3a://ka-app-code-<br/>&lt;user-name&gt; /data</b> |

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

## 編輯身分與存取權管

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 S3 儲存貯體許可

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 選擇政策。選擇主控台為您在上節所建立的 **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。使用您的帳戶 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 AWS Command Line Interface 以使用建立和執行 Apache Flink 應用程式的受管理服務。使用指 AWS CLI 令為 Apache Flink 應用程式建立受管理的服務，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**。使用您的帳戶 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. 前往 <https://console.aws.amazon.com/iam/> 開啟 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. 選擇下一步：許可。
7. 在連接許可政策頁面，選擇下一步：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，選擇許可標籤。
- b. 選擇連接政策。
- 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。使用您之前建立之角色的 ARN，取代範例角色 ARN。使用您在上一節中選擇的尾碼取代儲存貯體 ARN 尾碼 (username)。使用您的帳戶 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式：

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

應用程式現在已建立。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，您將使用[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": "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

在本節中，您可以使用[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，而無需重新編譯應用程式程式碼。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-/data"
          }
        }
      ]
    }
  }
}
```

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

```
aws kinesisanalyticstv2 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 [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。如需如何使用 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": "SAMPLEUehYngP87ex1nzYIGYgfhyppvDU"  
        }  
      }  
    }  
  }  
}
```

## 清理 AWS 資源

本節包括清除在「暫停視窗」教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

### 刪除適用於 Apache Flink 應用程式的受管理服務

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，選擇刪除，然後確認刪除。

### 刪除您的 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，選擇「ExampleOutputStream 動作」，選擇「刪除」，然後確認刪除。

### 刪除您的 Amazon S3 對象和存儲桶

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。

### 刪除您的 IAM 資源

1. 前往 <https://console.aws.amazon.com/iam/> 開啟 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-we st-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 us-west-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/) <https://console.aws.amazon.com/cloudwatch/>
2. 在導覽列中，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。

# 使用具有 Apache Flink 管理服務的工作室筆記本

適用於 Apache Flink 受管理服務的 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，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[教學課程：在 Apache Flink 的受管理服務中建立工作室筆記本](#)。如需 Apache Zeppelin 的詳細資訊，請參閱 [Apache Zeppelin 文件](#)。

使用筆記本，您可以使用 Apache Flink [表 API & SQL](#) 在 SQL，Python 或斯卡拉，或斯卡拉，或斯卡拉 [DataStream API](#) 中的模型查詢。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：

- [使用正確的 Studio 筆記本運行時版本](#)
- [建立工作室筆記本](#)
- [對串流資料執行互動式分析](#)
- [部署為具有持久狀態的應用程式](#)
- [檢閱 Studio 筆記本的 IAM 權限](#)
- [使用連接器和相依性](#)
- [實現用戶定義函](#)
- [啟用檢查點](#)
- [升級工作室運行](#)

- [使用 AWS Glue](#)
- [適用於 Apache Flink 管理服務中的工作室筆記本的範例和教學課程](#)
- [疑難排解適用於 Apache Flink 的受管理服務的工作室](#)
- [為 Apache Flink 工作室筆記本的受管理服務建立自訂IAM原則](#)

## 使用正確的 Studio 筆記本運行時版本

使用適用於 Apache Flink Studio 的 Amazon 受管服務，您可以在互動式筆記本中使用標準 SQL、Python 和 Scala 來實時查詢資料流，並建立和執行串流處理應用程式。工作室筆記本電腦是由[阿帕奇柏林飛艇](#)供電，並使用 [Apache Flink](#) 作為流處理引擎。

### Note

我們將在 2024 年 11 月 5 日棄用阿帕奇 Flink 版本 1.11 的工作室運行時。從這個日期開始，您將無法使用此版本執行新的筆記本或建立新的應用程式。我們建議您在此之前升級至最新的執行階段 (阿帕奇 Flink 1.15 和阿帕奇柏林飛艇 0.10)。如需如何升級筆記型電腦的指引，請參閱[升級工作室運行](#)。

### 工作室運行

| 阿帕奇 Flink 版本 | 阿帕奇飛艇版 | Python 版本 |               |
|--------------|--------|-----------|---------------|
| 1.15         | 0.10   | 3.8       | 建議            |
| 1.13         | 0.9    | 3.8       | 支援            |
| 1.11         | 0.9    | 3.7       | 於二〇二四年十一月五日棄用 |

## 建立工作室筆記本

Studio 筆記本包含用 Python 或 Scala 編寫的 SQL 查詢或程序，這些查詢或程序在流數據上運行並返回分析結果。您可以使用主控台或建立應用程式 CLI，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，例如亞馬遜MSK叢集、Kinesis 資料串流或 Amazon S3 儲存貯體。
  - 一個 AWS Glue 數據庫。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期屬性。如需應用程式設定和執行期屬性的相關資訊，請參閱 [Apache Flink 應用程式開發人員指南](#) 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 或建立您的 Studio 筆記本 AWS CLI。

從主控台建立應用程式時，您可以使用下列選項：

- 在 Amazon 主MSK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](#)。

## 對串流資料執行互動式分析

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[適用於 Apache Flink 管理服務中的工作室筆記本的範例和教學課程](#)下列內容和 Apache Flink 文件中的[查詢](#)。

您可以在 Studio 筆記本中使用 Flink SQL 查詢來查詢串流資料。您也可以使用 Python ( 表API ) 和 Scala ( 表和數據流APIs ) 編寫程序以交互方式查詢流數據。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          | 創建 ExecutionEnvironment//StreamExecutionEnvironmentBatchTableEnvironment/StreamTableEnvironment 並提供了一個 Scala 環境 |
| %flink.pyflink  | PyFlinkInterpreter        | 提供了一個蟒蛇環境   |
| %flink.ipyflink | IPyFlinkInterpreter       | 提供一個安全的環境   |
| %flink.ssql     | FlinkStreamSqlInterpreter | 提供一個流的 sql 環境   |
| %flink.bsqli    | FlinkBatchSqlInterpreter  | 提供一個批處理 sql 環境  |

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  |
|--------------------|---|
| <code>sekv</code>  | <code>StreamExecutionEnvironment</code>               |
| <code>stenv</code> | <code>StreamTableEnvironment for blink planner</code> |

您可以使用以下變數存取 Python 資料表環境資源：

| 變數                 | 資源  |
|--------------------|---|
| <code>s_kv</code>  | <code>StreamExecutionEnvironment</code>               |
| <code>st_kv</code> | <code>StreamTableEnvironment for blink planner</code> |

如需有關使用資料表環境的詳細資訊，請參閱 Apache Flink 文件API中的[概念和一般](#)。

## 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的一樣。您可以透過在筆記中選擇執行所有段落來檢查此順序。
- 你的代碼是 Python 和 SQL 或斯卡拉和 SQL。我們目前不支持 Python 和斯卡拉在一起 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。
- 至少必須存在一個含有原始程式碼 (Flink PyFlink 或 Flink Scala) 的段落 SQL，這些段落必須啟用此段落，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，您可以透過更新已部署的應用程式 AWS Management Console，AWS Command Line Interface 或 AWS API 根據應用程式所需的平行處理原則層級變更平行處理原則和/或 `ParallelismPerKPU` 設定，或者您可以為已部署的應用程式啟用自動調度資源。
- 如果您部署為具有持久狀態的應用程序，則 VPC 必須具有 Internet 訪問權限。如果您 VPC 沒有網際網路連線，請參閱 [在沒有網際網路存取的情況下部署為 VPC 具有持久狀態的應用程序](#)。

## 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。
- 您的 Python 段落不得使用 [shell 調用/分配](#) 使用 `!` 或 [IPython 魔術命令](#)，例如 `%timeit` 或 `%conda` 在註釋中將其部署為應用程序。

- 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。有關 Scala 案例類的信息，請參閱 Scala 文檔[CASECLASSES](#)中的。

## SQL 標準

- 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送資料。
- 在任何給定段落中，DDL陳述式 (USECREATE、ALTER、DROPSET、RESET) 必須在 DML (INSERT) 陳述式之前。這是因為段落中的DML陳述式必須一起提交為單一 Flink 工作。
- 最多應該有一個段落中有DML陳述。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，請參閱使用具有適用於 [Apache Flink](#)、[Amazon Translate](#) 和亞馬遜 [Amazon Comprehend](#) 的亞馬遜受管服務的SQL函數來 [Amazon Translate](#)、[編修和分析串流資料](#)。

## 檢閱 Studio 筆記本的IAM權限

當您透過建立 Studio 筆記本時，Apache Flink 的受管理服務會為您建立IAM角色。AWS Management Console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 |



## 使用連接器和相依性

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。如需有關連接器的詳細資訊，請參閱 Apache Flink 文件中的資料表[和SQL連接器](#)。

### 預設連接器

如果您使用建立 Studio 筆記本，Apache Flink 的受管理服務預設會包含下列自訂連接器：flink-sql-connector-kinesis、flink-connector-kafka\_2.12和aws-msk-iam-auth。AWS Management Console 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[CreateApplication](#) API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MavenReference，如下列範例所示。

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

#### 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[UpdateApplication](#) API 作業並將其指定為 `CustomArtifactsConfigurationUpdate` 資料類型 `MavenReference` 中的。

#### Note

您可以針對 `failOnError` 對表格中的 `flink-sql-connector-kinesis` 連接器設定為 `true` API。

## 新增相依性和自訂連接器

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

1. 將自訂連接器的檔案上傳到 Amazon S3。
2. 在中 AWS Management Console，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 Save changes (儲存變更)。

若要在使用建立新的 Studio 筆記本時新增相依性JAR或自訂連接器 [CreateApplication](#) API，請在 CustomArtifactsConfiguration 資料類型中指定相依性的 Amazon S3 位置JAR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 [UpdateApplication](#) API 作業，並在 CustomArtifactsConfigurationUpdate 資料類型中指定相依性的 Amazon S3 位置JAR或自訂連接器。

### 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。

## 實現用戶定義函

用戶定義函數 ( UDFs ) 是擴展點，允許您調用常用邏輯或自定義邏輯，這些邏輯不能以其他方式在查詢中表達。您可以使用 Python 或類似 Java 或斯卡拉的 JVM 語言來實現您的工作室筆記本UDFs中的段落。您也可以將包含以某種JVM語言UDFs實作的 Studio 筆記本外部JAR檔案加入。

當實現JARs該註冊子類UserDefinedFunction ( 或您自己的抽象類 ) 的抽象類時，請在 Apache Maven 中使用提供的範圍，Gradle 中的 compileOnly 依賴聲明 SBT，在中提供的範圍或UDF項目構建配置中的等效指令。這允許UDF源代碼針對 Flink 進行編譯APIs，但 Flink API 類本身不包含在構建工件中。從堅持 Maven 項目上這樣的先決條件的 UDF jar 例子中參考這個 [pom](#)。

### Note

如需設定範例，請參閱 Machine Learning 部落格上的 [使用 Amazon 受管服務的 Apache Flink、亞馬遜 Translate 和 Amazon Comprehend 使用SQL功能來翻譯、編輯和分析串流資料](#)。AWS

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

1. 將您的UDFJAR檔案上傳到 Amazon S3。
2. 在中 AWS Management Console，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 位置或具有您的UDF. ZIP
6. 選擇 Save changes (儲存變更)。

若要在使用建立新 Studio 筆記本UDFJAR時加入 [CreateApplication](#)API，請在CustomArtifactConfiguration資料類型中指定JAR位置。若要新增UDFJAR至現有 Studio 筆記本，請呼叫[UpdateApplication](#)API作業並指定CustomArtifactsConfigurationUpdate資料類型中的JAR位置。或者，您可以使用將UDFJAR檔案加入 AWS Management Console 至您的 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，在部署為應用程式選項期間，會透過將的主類別名稱轉換為小寫字串，JAR進行隱含函數註冊。

例如，如果TextAnalyticsUDF是的主類 UDFJAR，則隱式註冊將導致函數名稱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
;
```

- 如果您UDFJAR包含 FlinkSDKs，請配置您的 Java 項目，以便UDF源代碼可以針對 Flink 進行編譯 SDKs，但 Flink SDK 類本身不包含在構建工件中，例如. JAR

您可以在 Apache Maven 中使用provided範圍，Gradle 中的compileOnly依賴聲明，provided範圍或其UDF項目構建配置中的等效指令。SBT您可以從 UDF jar 示例中引用此 [pom](#)，該示例堅持 maven 項目上的先決條件。如需完整的 step-by-step 教學課程，請參閱這篇文章，[使用具有適用於 Apache Flink、Amazon Translate 和亞馬遜 Amazon Comprehend 的亞馬遜受管服務的SQL功能來翻譯、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](#)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"
)
```

## 升級工作室運行

本節包含有關如何升級您的 Studio 筆記本運行時的信息。我們建議您一律升級至最新支援的 Studio 執行階段。

### 將您的筆記本升級到新的 Studio 運行時

根據您使用 Studio 的方式，升級執行階段的步驟會有所不同。選取適合您使用案例的選項。

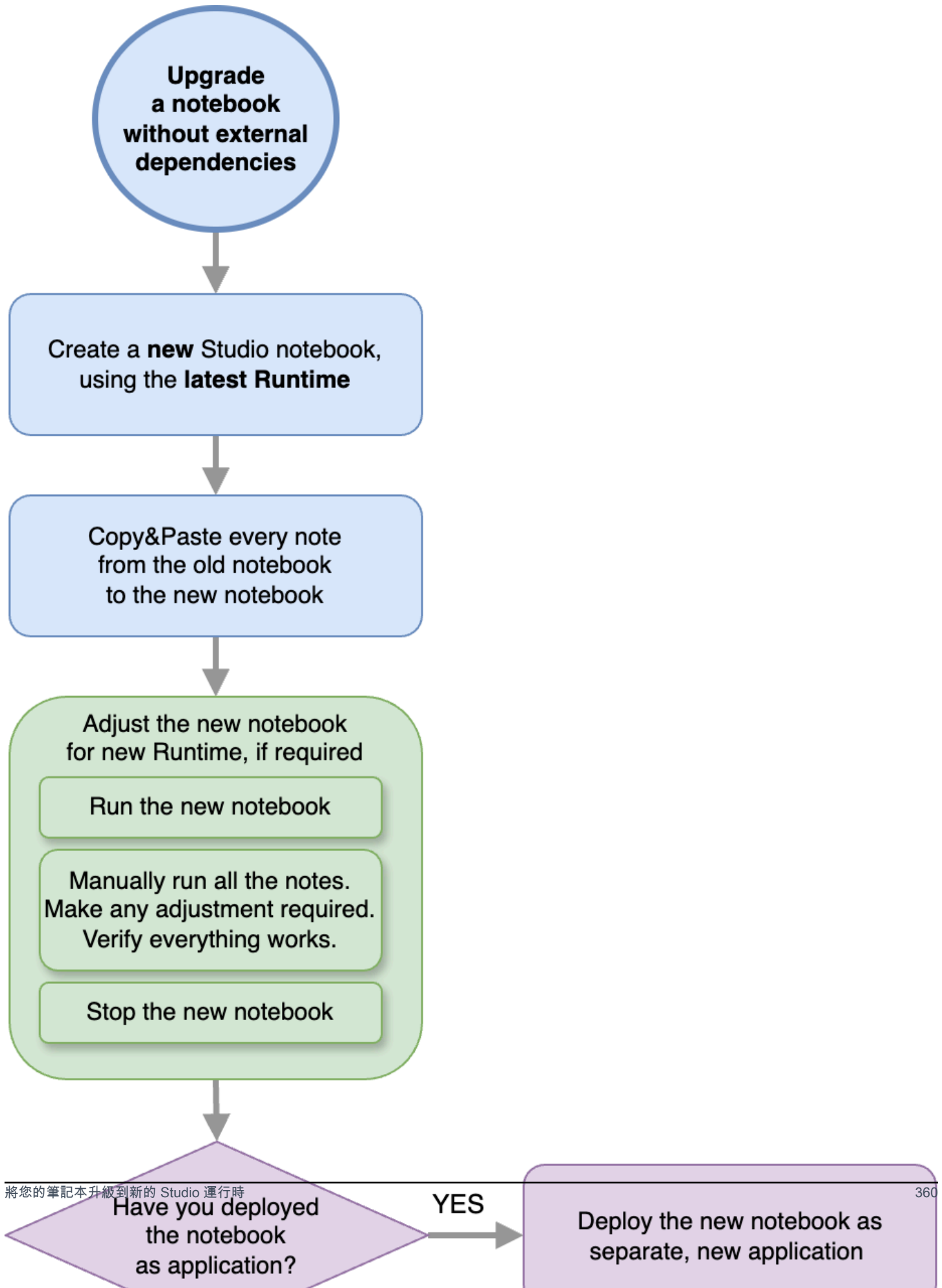
#### SQL查詢或沒有外部依賴關係的 Python 代碼

如果您使用SQL或沒有任何外部相依性的 Python，請使用下列執行階段升級程序。我們建議您升級至最新的執行階段版本。升級程序與您要從中升級的執行階段版本相同，無後端。

1. 使用最新的執行階段建立新的 Studio 筆記本。
2. 將舊筆記本中每個記事的代碼複製並粘貼到新筆記本中。
3. 在新筆記本中，調整程式碼，使其與舊版變更的任何 Apache Flink 功能相容。
  - 執行新的筆記本。打開筆記本並按照筆記順序運行它，並測試它是否有效。
  - 對程式碼進行任何必要的變更。
  - 停止新的筆記本。
4. 如果您已將舊筆記型電腦部署為應用程式：
  - 將新的筆記本部署為單獨的新應用程式。
  - 停止舊的應用程式。

- 執行不含快照的新應用程式。
5. 如果舊筆記本正在運行，請停止它。視需要啟動新筆記本以供互動使用。

無需外部依賴關係即可升級的流程



將您的筆記本升級到新的 Studio 運行時

Have you deployed the notebook as application?

YES

Deploy the new notebook as separate, new application

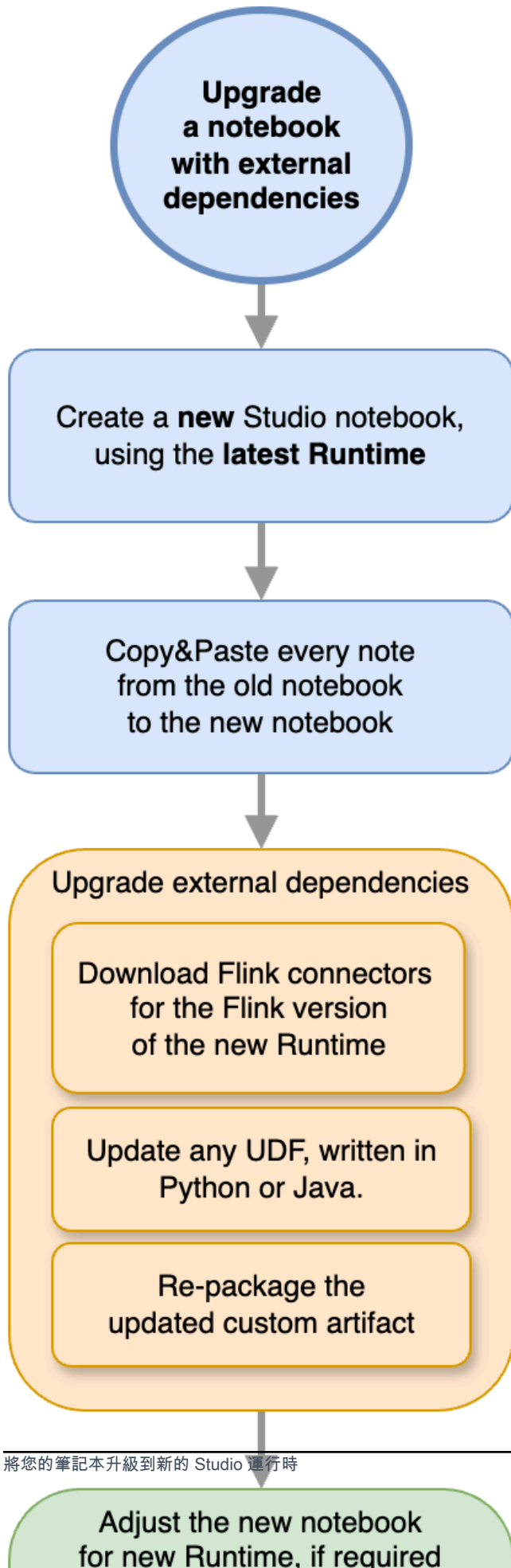


## SQL具有外部依賴關係的查詢或 Python 代碼

如果您正在使用SQL或 Python 並使用外部依賴項（例如連接器或自定義成品），例如在 Python 或 Java 中實現的用戶定義函數，請遵循此過程。我們建議您升級至最新的執行階段。無論您要從哪個執行階段版本升級，程序都是相同的。

1. 使用最新的執行階段建立新的 Studio 筆記本。
2. 將舊筆記本中每個記事的代碼複製並粘貼到新筆記本中。
3. 更新外部相依性和自訂人工因素。
  - 尋找與新執行階段 Apache Flink 版本相容的新連接器。請參閱 Apache Flink 文件中的[資料表和 SQL連接器](#)，以尋找 Flink 版本的正確連接器。
  - 更新使用者定義函數的程式碼，以符合 Apache Flink 中的變更API，以及使用者定義函數使用的任何 Python 或JAR相依性。重新封裝您更新的自訂成品。
  - 將這些新的連接器和成品新增至新的筆記本。
4. 在新筆記本中，調整程式碼，使其與舊版變更的任何 Apache Flink 功能相容。
  - 執行新的筆記本。打開筆記本並按照筆記順序運行它，並測試它是否有效。
  - 對程式碼進行任何必要的變更。
  - 停止新的筆記本。
5. 如果您已將舊筆記型電腦部署為應用程式：
  - 將新的筆記本部署為單獨的新應用程式。
  - 停止舊的應用程式。
  - 執行不含快照的新應用程式。
6. 如果舊筆記本正在運行，請停止它。視需要啟動新筆記本以供互動使用。

### 使用外部相依性升級的程序流程



## 使用 AWS Glue

您的 Studio 筆記本會儲存並從中取得其資料來源和接收器的相關資訊 AWS Glue。建立 Studio 筆記本時，請指定包含連線資訊的資 AWS Glue 料庫。存取資料來源和接收器時，您可以指定資料庫中包含的 AWS Glue 表格。您的 AWS Glue 表格可讓您存取定義資料來源和目標的位置、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 AWS Glue 連線、資料庫和表格以搭配 Studio 筆記本使用的範例，請參閱自學課程[建立資 AWS Glue 料庫](#)中的[教學課程：在 Apache Flink 的受管理服務中建立工作室筆記本](#) 〈〉。

### 資料表屬性

除了資料欄位之外，您的資料 AWS Glue 表還會使用表格屬性為您的 Studio 筆記本提供其他資訊。適用於 Apache Flink 的受管理服務會使用下列 AWS Glue 表格屬性：

- [定義阿帕奇 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 連接器和格式內容](#)：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 表格，請執行下列操作：

1. 登入 AWS Management Console 並開啟 AWS Glue 主控台，位於<https://console.aws.amazon.com/glue/>。
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**。

### 定義阿帕奇 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，請在 AWS Glue 資料表上定義屬性，以告知 Apache Flink 執行階段的受管理服務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                                      |
|----------------------|---|--|
| <a href="#">處理時間</a> | 管理 FLINK                                    | AWS Glue 將用於暴露值的列名稱。此欄名稱不對應於現有的資料表資料行。 |
| <a href="#">活動時間</a> | 管理弗林克. 行時間                                  | AWS Glue 將用於暴露值的列名稱。此欄名稱對應於現有的表格欄。     |
|                      | 管理 FLINK. 水印。 <code>column_name</code> . 毫秒 | 浮水印間隔 (毫秒)                             |

## 使用 Flink 連接器和格式內容

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   |
|-----------------------------|-------------------|---|
| <a href="#">卡夫卡</a>         | format            | 用於反序列化和序列化卡夫卡消息的格式，例如或。json csv           |
|                             | scan.startup.mode | 卡夫卡消費者的啟動模式，例如或 earliest-offset timestamp |
| <a href="#">Kinesis</a>     | format            | 用來反序列化和序列化 Kinesis 資料串流記錄的格式，例如或。json csv |
|                             | aws.region        | 定義串流的 AWS 區域。                             |
| <a href="#">S3 ( 文件系統 )</a> | 格式                | 用於反序列化和序列化文件的格式，例如或。json csv              |

|       |      |                                |
|-------|------|--------------------------------|
| 連接器類型 | 金鑰   | 值                              |
|       | path | Amazon S3 路徑，例如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。

## 適用於 Apache Flink 管理服務中的工作室筆記本的範例和教學課程

### 主題

- [教學課程：在 Apache Flink 的受管理服務中建立工作室筆記本](#)
- [教學課程：將 Studio 筆記本部署為具有持久狀態的 Apache Flink 應用程式的受管理服務](#)
- [查看示例查詢以分析 Studio 筆記本中的數據](#)

### 教學課程：在 Apache Flink 的受管理服務中建立工作室筆記本

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：

- [完成先決條件](#)
- [建立資 AWS Glue 料庫](#)
- [後續步驟：使用 Kinesis Data Streams 或 Amazon 建立工作室筆記本 MSK](#)
- [使用 Kinesis Data Streams 建立 Studio 筆記本](#)
- [使用 Amazon 創建工作室筆記本 MSK](#)
- [清理您的應用程式和相依資源](#)

### 完成先決條件

確保您的版本 AWS CLI 是 2 或更高版本。若要安裝最新版本 AWS CLI，請參閱[安裝、更新和解除安裝 AWS CLI 版本 2](#)。

### 建立資 AWS Glue 料庫

您的 Studio 筆記本使用資料[AWS Glue](#)庫取得有關 Amazon MSK 資料來源的中繼資料。

## 建立資 AWS Glue 料庫

1. 在開啟 AWS Glue 主控台<https://console.aws.amazon.com/glue/>。
2. 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 Create (建立)。

## 後續步驟：使用 Kinesis Data Streams 或 Amazon 建立工作室筆記本 MSK

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

- [使用 Kinesis Data Streams 建立 Studio 筆記本](#)：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。
- [使用 Amazon 創建工作室筆記本 MSK](#)：使用 AmazonMSK，您可以建立使用 Amazon MSK 叢集做為來源的應用程式。您需要創建一個 AmazonVPC，Amazon EC2 客戶端實例和一個 Amazon MSK 集群作為相關資源。

## 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：

- [完成先決條件](#)
- [創建一個 AWS Glue 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](#)

### 完成先決條件

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**。

若要使用建立串流 (ExampleInputStream) AWS CLI，請使用下列 Amazon Kinesis create-stream AWS CLI 命令。

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

## 創建一個 AWS Glue 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。同樣地，您可以依照本節所述手動建立表格，也可以在 Apache Zeppelin 的筆記本中，使用為 Apache Flink 管理服務建立表格連接器程式碼，透過陳述式建立表格。DDL 然後，您可以入庫 AWS Glue 以確保表格已正確建立。

## 建立資料表

1. 登入 AWS Management Console 並開啟 AWS Glue 主控台，位於<https://console.aws.amazon.com/glue/>。
2. 如果您還沒有資料 AWS Glue 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 Create (建立)。
3. 在左側導覽列中，選擇資料表。在資料表頁面，選擇新增資料表 > 手動新增資料表。
4. 在設定資料表頁面，為資料表名稱輸入 **stock**。請務必選取先前建立的資料庫。選擇 Next (下一步)。
5. 在新增資料存放區頁面，選擇 Kinesis。對於串流名稱，輸入 **ExampleInputStream**。針對 Kinesis 來源 URL，選擇 Enter **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 Next (下一步)。
6. 在「分類」頁面中，選擇 JSON。選擇 Next (下一步)。
7. 在定義結構描述頁面，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          |
|---------------|---------------|
| <b>ticker</b> | <b>string</b> |

| 欄名稱          | 資料類型          |
|--------------|---------------|
| <b>price</b> | <b>double</b> |

選擇 Next (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。

若要建立您的應用程式，您可以使用 AWS Management Console 或 AWS CLI。

- [建立工作室筆記本 AWS Management Console](#)
- [建立工作室筆記本 AWS CLI](#)

## 建立工作室筆記本 AWS Management Console

1. 在 <https://console.aws.amazon.com/managed-flink/> 家中打開 Apache Flink 控制台的託管服務？  
[區域 = us-east-1 #/應用程式/儀表板](#)。
2. 在 Managed Service for Apache Flink 應用程式頁面，選擇 Studio 標籤。選擇建立 Studio 筆記本。

### Note

您也可以透過選取輸入的 Amazon MSK 叢集MSK或 Kinesis 資料串流，然後選擇即時處理資料，從 Amazon 或 Kinesis 資料串流主控台建立 Studio 筆記本。

3. 在建立 Studio 筆記本頁面，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。

### 建立工作室筆記本 AWS CLI

若要使用建立您的 Studio 筆記本 AWS CLI，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/ZeppeleinRole`，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。使用您的資訊取代預留位置的值。

```
{
  "ApplicationName": "MyNotebook",
  "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
  "ApplicationMode": "INTERACTIVE",
  "ServiceExecutionRole": "arn:aws:iam::AccountID:role/ZeppeleinRole",
  "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 kinesisanalyticstv2 create-application --cli-input-json file://create.json
```

- 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:kinesisanalyticstv2-east-1:012345678901:application/MyNotebook",
    "ApplicationName": "MyNotebook",
    "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
    "ApplicationMode": "INTERACTIVE",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/ZepppelinRole",
    ...
  }
}
```

- 若要執行應用程式，請執行下列命令：使用您的帳戶 ID 取代範例值。

```
aws kinesisanalyticstv2 start-application --application-arn
arn:aws:kinesisanalyticstv2-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：

- 開啟 [Kinesis 資料產生器](#)。
- 選擇使用建立 Cognito 使用 CloudFormation 者。
- AWS CloudFormation 主控台隨即開啟，其中包含 Kinesis 資料產生器範本。選擇 Next (下一步)。
- 在指定堆疊詳細資訊頁面，輸入 Cognito 使用者的使用者名稱和密碼。選擇 Next (下一步)。
- 在設定堆疊選項頁面，選擇下一步。
- 在 [檢閱運動資料產生器-認知-使用者] 頁面中，選擇 [我確認可能會建立資源的]。AWS CloudFormation IAM 核取方塊。選擇 Create Stack (建立堆疊)。
- 等待 AWS CloudFormation 堆棧完成創建。堆疊完成後，在主控台中開啟 [運動資料產生器-齒狀-使用者] 堆疊，然後選擇 [輸出] 索引標籤。AWS CloudFormation 打開 URL 列出的 KinesisDataGeneratorUrl 輸出值。
- 在 Amazon Kinesis 資料產生器頁面，使用您在步驟 4 中建立的憑證登入。
- 在下一頁面，提供下列值：

|              |                           |
|--------------|---------------------------|
| 區域           | <b>us-east-1</b>          |
| 流/Firehose 溪 | <b>ExampleInputStream</b> |
| 每秒記錄         | <b>1</b>                  |

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. 在 <https://console.aws.amazon.com/managed-flink/家中打開 Apache Flink 控制台的託管服務？區域 = us-east-1 # /應用程式/儀表板>。
2. 在 Managed Service for Apache Flink 應用程式頁面，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，選擇 Zeppelin 筆記。

5. 在 Zeppelin 筆記頁面，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。

若要為您的應用程式開啟 Apache Flink 儀表板以檢視操作層面，請選擇 FLINKJOB。如需 Flink 儀表板的詳細資訊，請參閱 [Managed Service for Apache Flink 開發人員指南](#) 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](#) 文件中的 [查詢](#)。

## 使用 Amazon 創建工作室筆記本 MSK

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

本教學課程包含下列章節：

- [設置 Amazon MSK 群集](#)
- [將 NAT 閘道新增至您的 VPC](#)
- [創建一個 AWS Glue 連接和表](#)
- [使用 Amazon 創建工作室筆記本 MSK](#)
- [將資料傳送到您的 Amazon MSK 叢集](#)
- [測試 Studio 筆記本](#)

### 設置 Amazon MSK 群集

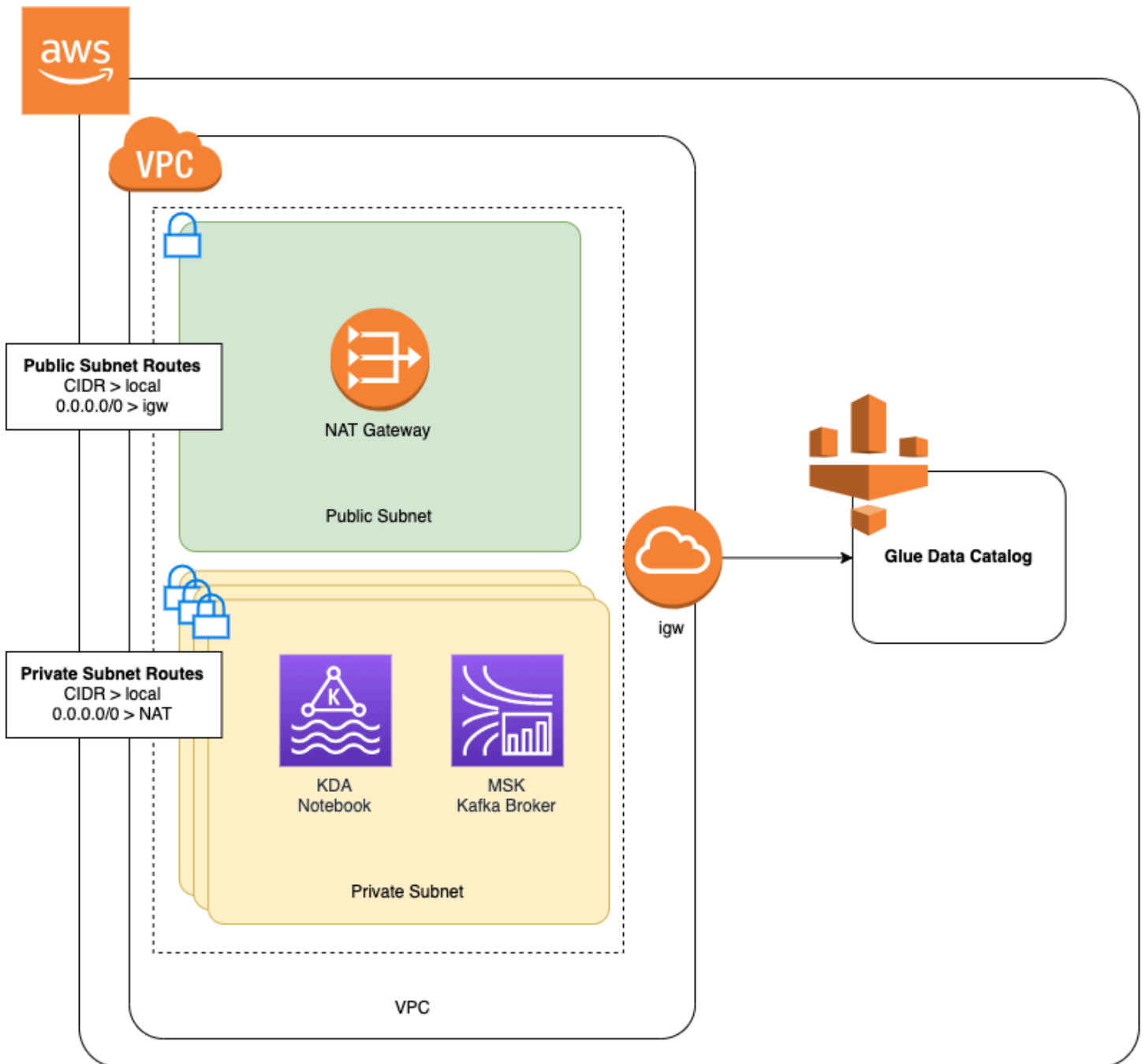
在本教學中，您需要一個允許純文字存取的 Amazon MSK 叢集。如果您尚未設定 Amazon MSK 叢集，請按照 [Amazon 入門使用 MSK](#) 教學建立 Amazon VPC、Amazon MSK 叢集、主題和 Amazon 用 EC2 戶端執行個體。

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](#) 中，在步驟 4 中，將 ClientBroker 值從變更 TLS 為 **PLAINTEXT**。

## 將NAT閘道新增至您的 VPC

如果您按照[開始使用 Amazon MSK 教學建立 Amazon MSK 叢集](#)，或者您現有的 Amazon VPC 未擁有其私有子網路的 NAT 閘道，則必須將 NAT 閘道新增到 Amazon VPC。下圖顯示一般架構。



要為您的 Amazon 創建 NAT 網關 VPC，請執行以下操作：

1. 在打開 Amazon VPC 控制台 <https://console.aws.amazon.com/vpc/>。
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-可選

**ZeppelinGateway**

子網

AWS KafkaTutorialSubnet1

彈性 IP 配置識別碼

選擇可用的彈性 IP。如果沒有IPs可用的彈性，請選擇 [配置彈性 IP]，然後選擇主控台建立的 Elastic IP。

選擇 [建立NAT閘道]。

5. 在導覽列中，選擇路由表。
6. 選擇建立路由表。
7. 在建立路由表頁面，提供以下資訊：

- 名稱標籤：**ZeppelinRouteTable**
- VPC：選擇您的VPC ( 例如 AWS KafkaTutorialVPC )。

選擇 Create (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**。針對「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，選取後，ZeppelinRouteTable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 [AWS KafkaTutorialSubnet2] 和 [AWS KafkaTutorialSubnet3]。選擇 Save (儲存)。

## 創建一個 AWS Glue 連接和表

您的 Studio 筆記本使用資料[AWS Glue](#)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 並開啟 AWS Glue 主控台，位於<https://console.aws.amazon.com/glue/>。
2. 如果您還沒有資料 AWS Glue 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 Create (建立)。
3. 從左側導覽列選擇連線。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 引導服務器 URLs，為您的集群提供引導代理字符串。您可以從MSK控制台或輸入以下CLI命令來獲取引導代理程序：

```
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 Next (下一步)。

5. 在VPC頁面中，提供下列值：
  - 對於 VPC，選擇您的名稱VPC ( 例如 AWS KafkaTutorialVPC。 )
  - 對於子網路，選擇 AWS KafkaTutorialSubnet2。
  - 對於安全群組，請選擇所有可用的群組。

選擇 Next (下一步)。

6. 在連線屬性 / 連線存取權頁面，選擇完成。

## 建立資料表

### Note

您可以依照下列步驟中所述手動建立表格，也可以在 Apache Zeppelin 的筆記本中，使用針對 Apache Flink 的受管理服務建立表格連接器程式碼，透過陳述式建立表格。DDL然後，您可以入庫 AWS Glue 以確保表格已正確建立。

1. 在左側導覽列中，選擇資料表。在資料表頁面，選擇新增資料表 > 手動新增資料表。
2. 在設定資料表頁面，為資料表名稱輸入 **stock**。請務必選取先前建立的資料庫。選擇 Next (下一步)。
3. 在新增資料存放區頁面，選擇 Kafka。對於主題名稱，輸入您的主題名稱 (例如 AWS KafkaTutorialTopic)。對於「連線」，請選擇ZeppelinConnection。
4. 在「分類」頁面中，選擇JSON。選擇 Next (下一步)。
5. 在定義結構描述頁面，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          |
|---------------|---------------|
| <b>ticker</b> | <b>string</b> |
| <b>price</b>  | <b>double</b> |

選擇 Next (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「編輯表格」並新增下列內容：
  - 鍵：managed-flink.proctime，值：proctime
  - 鍵：flink.properties.group.id，值：test-consumer-group
  - 鍵：flink.properties.auto.offset.reset，值：latest
  - 鍵：classification，值：json

如果沒有這些鍵/值對，Flink 筆記本會遇到錯誤。

9. 選擇套用。

使用 Amazon 創建工作室筆記本 MSK

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 或建立應用程式 AWS CLI。

- [建立工作室筆記本 AWS Management Console](#)
- [建立工作室筆記本 AWS CLI](#)



**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。

### 建立工作室筆記本 AWS Management Console

1. 在 <https://console.aws.amazon.com/managed-flink/>家中打開 Apache Flink 控制台的託管服務？區域 = us-east-1 #/應用程式/儀表板。
2. 在 Managed Service for Apache Flink 應用程式頁面，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，請選取您輸入的 Amazon MSK 叢集或 Kinesis 資料串流，然後選擇「即時處理資料」。

3. 在建立 Studio 筆記本頁面，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」頁籤。在網路模式區段中，選擇編輯。
5. 在 [編輯以下項目的聯網 MyNotebook] 頁面中，選擇以 Amazon MSK 叢集為基礎的 VPC 組態。為 Amazon MSK 群集選擇您的 Amazon MSK 群集。選擇 Save changes (儲存變更)。
6. 在 MyNotebook 頁面中，選擇 [執行]。等待狀態顯示為執行中。

### 建立工作室筆記本 AWS CLI

若要使用建立您的 Studio 筆記本 AWS CLI，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。
  - 包含您的 Amazon MSK 叢集之 Amazon VPC 的子網路 IDs 和安全群組識別碼。

2. 建立稱為 `create.json` 的檔案，其中具有以下內容。使用您的資訊取代預留位置的值。

```
{
  "ApplicationName": "MyNotebook",
  "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
  "ApplicationMode": "INTERACTIVE",
  "ServiceExecutionRole": "arn:aws:iam::AccountID:role/ZepppelinRole",
  "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 kinesisanalyticstv2 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. 若要執行應用程式，請執行下列命令：使用您的帳戶 ID 取代範例值。

```
aws kinesisanalyticstv2 start-application --application-arn
arn:aws:kinesisanalyticsus-east-1:012345678901:application/MyNotebook\
```

### 將資料傳送到您的 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. Connect 到您的 Amazon EC2 客戶端。
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，AWS CLI 在用戶端電腦上設定：

```
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。將範例值取代為 Amazon MSK 叢集的 Bootstrap 代理程式字串，如果您的主題不是，請更新主題名稱 `AWS KafkaTutorialTopic`：

```
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. 在 <https://console.aws.amazon.com/managed-flink/> 家中打開 Apache Flink 控制台的託管服務？  
[區域 = us-east-1 # /應用程式/儀表板](#)。
2. 在 Managed Service for Apache Flink 應用程式頁面，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，選擇 Zeppelin 新筆記。
5. 在 Zeppelin 筆記頁面，在新筆記中輸入以下查詢：

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

選擇執行圖示。

應用程式會顯示來自 Amazon MSK 叢集的資料。

若要為您的應用程式開啟 Apache Flink 儀表板以檢視操作層面，請選擇 FLINKJOB。如需 Flink 儀表板的詳細資訊，請參閱 [Managed Service for Apache Flink 開發人員指南](#) 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](#) 文件中的 [查詢](#)。

## 清理您的應用程式和相依資源

### 刪除 Studio 筆記本

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

### 刪除您的 AWS Glue 數據庫和連接

1. 在開啟 AWS Glue 主控台 <https://console.aws.amazon.com/glue/>。
2. 從左側導覽列選擇資料庫。核取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。選中旁邊的複選框 ZeppelinConnection 以選擇它。依序選擇動作和刪除連線。確認您的選擇。

## 刪除您的IAM角色和政策

1. 在開啟IAM主控台<https://console.aws.amazon.com/iam/>。
2. 從左側導覽選單選擇角色。
3. 使用搜尋列來搜尋ZeppelinRole角色。
4. 選擇ZeppelinRole角色。選擇刪除角色。確認刪除。

## 刪除您的 CloudWatch 日誌群組

當您使用主控台建立應用程式時，主控台會為您建立 CloudWatch 記錄群組和記錄資料流。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. 在開啟 CloudWatch 主控台<https://console.aws.amazon.com/cloudwatch/>。
2. 從左側導覽列選擇日誌群組。
3. 選擇//AWSKinesisAnalytics/MyNotebook記錄群組。
4. 選擇動作、刪除日誌群組。確認刪除。

## 清理 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

如果您為本教學建立 Amazon MSK 叢集，請遵循本節中的步驟。本節提供清理 Amazon EC2 客戶端實例VPC，Amazon 和 Amazon MSK 集群的說明。

## 刪除您的 Amazon MSK 群集

如果您針對本教學建立 Amazon MSK 叢集，請遵循下列步驟。

1. 在<https://console.aws.amazon.com/msk/>家打開 Amazon MSK 控制台？區域 = us-east-1 #/ 家/。
2. 選擇AWS KafkaTutorialCluster。選擇 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

如果您為本教學建立 Amazon EC2 用戶端執行個體，請遵循下列步驟。

1. 在打開 Amazon EC2 控制台<https://console.aws.amazon.com/ec2/>。

2. 從左側導覽列中選擇執行個體。
3. 選擇旁邊的複選框ZeppelinClient以將其選中。
4. 依序選擇執行個體狀態和終止執行個體。

## 刪除您的 Amazon VPC

如果您VPC為本教程創建了 Amazon ，請按照以下步驟操作。

1. 在打開 Amazon EC2 控制台<https://console.aws.amazon.com/ec2/>。
2. 從左側導覽列選擇網路介面。
3. 在搜索欄中輸入您的 VPC ID ，然後按 Enter 鍵進行搜索。
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. 在打開 Amazon VPC 控制台<https://console.aws.amazon.com/vpc/>。
9. 選取AWS KafkaTutorialVPC。選擇動作，刪除VPC。輸入 **delete** 並確認刪除。

## 教學課程：將 Studio 筆記本部署為具有持久狀態的 Apache Flink 應用程式的受管理服務

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：

- [完成事前準備](#)
- [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](#)

### 完成事前準備

按照使用 Kinesis Data Streams 或 Amazon [教學課程：在 Apache Flink 的受管理服務中建立工作室筆記本MSK](#)，建立新的 Studio 筆記本。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：
  - 亞馬遜 FullAccess
  - 亞馬遜管理 flinkFullAccess
  - 訪問您的來源，目的地和適VPCs用的。如需詳細資訊，請參閱[檢閱 Studio 筆記本的IAM權限](#)。
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。



- 這將引導您進入用於為 Apache Flink 應用程式創建託管服務的 AWS Management Console 頁面。請注意，已預先填入應用程式名稱、平行處理原則、程式碼位置、預設 Glue DB VPC (如果適用) 和 IAM 角色。驗證這些 IAM 角色具有來源和目的地的必要權限。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

若要使用部署應用程式 AWS CLI，您必須更新您的，AWS CLI 才能使用隨 Beta 2 資訊提供的服務模型。如需如何使用更新的服務模型之相關資訊，請參閱 [完成先決條件](#)。

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 kinesisanalyticstv2 start-application \
  --application-name <app-name> \
  --region us-east-1 \
  --no-verify-ssl

```

下列程式碼會URL傳回應用程式的 Apache 齊柏林飛艇筆記本頁面：

```

aws kinesisanalyticstv2 create-application-presigned-url \
  --application-name <app-name> \
  --url-type ZEPPELIN_UI_URL \

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

```

## 查看示例查詢以分析 Studio 筆記本中的數據

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。

- [MSK創建與亞馬遜/阿帕奇卡表格](#)
- [使用 Kinesis 動創建表](#)
- [查詢翻滾視窗](#)
- [查詢滑動視窗](#)
- [使用互動式 SQL](#)
- [使用 BlackHole SQL連接器](#)

- [使用斯卡拉生成示例數據](#)
- [使用互動斯卡拉](#)
- [使用互動式 Python](#)
- [使用交互式 Python，SQL和斯卡拉的組合](#)
- [使用跨帳戶 Kinesis 資料串流](#)

如需有關 Apache Flink SQL 查詢設定的資訊，請參閱用於互動式資料分析的[齊柏林飛艇筆記本上的 Flink](#)。

若要在 Apache Flink 儀表板中檢視您的應用程式，請 FLINKJOB 在應用程式的齊柏林飛艇注意頁面中選擇。

如需視窗查詢的詳細資訊，請參閱 [Apache Flink 文件](#) 中的 [視窗](#)。

如需 Apache Flink 串流 SQL 查詢的更多範例，請參閱 [Apache Flink](#) 文件中的 [查詢](#)。

## MSK 創建與亞馬遜/阿帕奇卡表格

您可以使用 Amazon MSK Flink 連接器搭配 Apache Flink 工作室的受管服務來驗證您的連線與純文字或身分驗證。SSL IAM 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 卡夫卡SQL](#) 連接器的其他屬性相結合。

## 使用 Kinesis 動創建表

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](#)。

## 查詢翻滾視窗

下列 Flink 串流SQL查詢會從表格中選取每個五秒鐘的翻滾視窗中的最高價格：ZeppelinTopic

```
%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)
```

## 查詢滑動視窗

下列 Apache Flink 串流SQL查詢會從表格中選取每五秒滑動視窗中的最高價格：ZeppelinTopic

```
%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)
```

## 使用互動式 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 “使用斯卡拉生成示例數據”](#) 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 文件中的文件：[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`;
```

## 使用 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。如需有關 BlackHoleSQL連接器的資訊，請參閱 Apache Flink 文件中的連[BlackHole SQL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
```

## 使用斯卡拉生成示例數據

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`
```

## 使用互動斯卡拉

這是 [the section called “使用互動式 SQL”](#) 的 Scala 翻譯。有關更多斯卡拉示例，請參閱 Apache Flink 文檔API中的[表](#)。

```
%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`
```

## 使用互動式 Python

這是 [the section called “使用互動式 SQL”](#) 的 Python 翻譯。如需更多 Python 範例，請參閱 Apache Flink 文件API中的[資料表](#)。

```
%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`
```

## 使用交互式 Python , SQL和斯卡拉的組合

您可以在筆記本中使用 Python 和 Scala 的SQL任何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序的 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!")
```

## 使用跨帳戶 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。在建立資料表DDL陳述`aws.credentials.role.sessionName`式的 Kinesis 連接器中使`aws.credentials.provider`用`aws.credentials.role.arn`、和，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.sql
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'
)
```

## 疑難排解適用於 Apache Flink 的受管理服務的工作室

本節包含 Studio 筆記本的疑難排解資訊。

### 停止卡住的應用程式

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](#) 中的 [執行應用程式](#)。

### 在沒有網際網路存取的情況下部署為 VPC 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 應用程式。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

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

5. 您可能需要為應用程式使用的服務授與 Apache Flink 工作室的受管理服務和 Apache Flink IAM 角色的受管理服務。您可以對這兩個應用程式使用相同的IAM角色。

## D deploy-as-app 尺寸和構建時間縮短

Python 應用程式的工作室 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 應用程式，如果您的應用程式不依賴，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，以避免違反應用程式大小的服務限制，還可以縮短具有該 deploy-as-app 功能的應用程式的構建時間。

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性。

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 應用程式之前刪除 **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
```

## 取消工作

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)
```

## 重新啟動阿帕奇 Flink 解釋器

在 Studio 筆記本中重新啟動 Apache Flink 解釋器

1. 選擇螢幕右上角附近的組態。
2. 選擇解釋器。
3. 選擇重新啟動，然後按確定。

## 為 Apache Flink 工作室筆記本的受管理服務建立自訂IAM原則

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

### Note

在下列政策範例中，使用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 集群](#)

## AWS Glue

下列範例原則會授與存取 AWS Glue 資料庫的權限。

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

## CloudWatch 日誌

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。

下列政策授與從用作來源的 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/*"
  }
]
}

```

下列政策授與向用作目的地的 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，則必須授與額外的許可，以存取該串流及其加密金鑰。

下列政策授與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>"
  ]
},

```

下列政策授與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 集群

若要授與 Amazon MSK 叢集的存取權，您可以授與叢集的存取權 VPC。如需存取 Amazon 的政策範例 VPC，請參閱[VPC 應用程式許可](#)。

# 開始使用阿帕奇 Flink 的 Amazon 託管服務 () DataStream API

本節將向您介紹 Apache Flink 受管理服務的基本概念，以及使用 DataStream API。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

- [檢閱適用於 Apache Flink 應用程式的受管理服務元件](#)
- [滿足完成練習的先決條件](#)
- [設定 AWS 帳戶並建立管理員使用者](#)
- [設置 AWS Command Line Interface \( AWS CLI \)](#)
- [為 Apache Flink 應用程式建立並執行受管理的服務](#)
- [清理 AWS 資源](#)
- [探索其他資源](#)

## 檢閱適用於 Apache Flink 應用程式的受管理服務元件

### Note

阿帕奇 Flink 的 Amazon 託管服務支持所有 Apache Flink APIs 和潛在的所有 JVM 語言。如需詳細資訊，請參閱 [Flink 的 APIs](#)。

視 API 您選擇的項目而定，應用程式的結構和實作會稍有不同。本入門教學課程涵蓋了 DataStream API 在 Java 中使用的應用程式的實作。

為了處理資料，Apache Flink 應用程式的受管理服務會使用 Java 應用程式，該應用程式會使用 Apache Flink 執行階段處理輸入並產生輸出。

Apache Flink 應用程式的典型託管服務具有以下組件：

- 執行階段屬性：您可以使用執行階段屬性，將組態參數傳遞至應用程式，以變更它們，而不需要修改和重新發佈程式碼。

- 來源：應用程式消耗來自一個或多個來源的數據。來源會使用[連接器](#)從外部系統讀取資料，例如 Kinesis 資料串流或 Kafka 值區。如需詳細資訊，請參閱[將串流資料來源新增至 Apache Flink 的受管理服務](#)。
- 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[在 Apache Flink 的受管理服務中使用運算子轉換資料](#)。
- 接收器：應用程式通過接收器將數據發送到外部源。接收器使用[連接器](#) v 將資料傳送到 Kinesis 資料串流、卡夫卡主題、Amazon S3 或關聯式資料庫。您也可以使用特殊的連接器列印輸出，僅用於開發目的。如需詳細資訊，請參閱[在 Apache Flink 的受管理服務中使用接收器寫入資料](#)。

您的應用程式需要一些外部相依性，例如應用程式使用的 Flink 連接器，或可能是 Java 程式庫。若要在適用於 Apache Flink 的亞馬遜受管服務中執行，必須將應用程式與相依性一起封裝在 fat-jar 中，然後上傳到 Amazon S3 儲存貯體。然後建立 Managed Service for Apache Flink 應用程式。您可以傳遞程式碼套件的位置，以及任何其他執行階段設定參數。

本教程演示如何使用 Apache Maven 來打包應用程式，以及如何在您選擇IDE的本地運行應用程式。

## 滿足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：

- [Git 用戶端](#)。如果您尚未安裝 Git 用戶端，請安裝。
- [Java 開發工具包 \(JDK\) 版本 11](#)。安裝 Java JDK 11 並將 JAVA\_HOME 環境變數設定為指向您的 JDK 安裝位置。如果你沒有一個 JDK 11，你可以使用 [Amazon Corretto 11](#) 或您選擇 JDK 的任何其他標準。
  - 若要確認您已正確JDK安裝，請執行下列命令。如果您使用的是 Amazon Corretto 以外的JDK輸出將有所不同。確保該版本是 11.x。

```
$ java --version

openjdk 11.0.23 2024-04-16 LTS
OpenJDK Runtime Environment Corretto-11.0.23.9.1 (build 11.0.23+9-LTS)
OpenJDK 64-Bit Server VM Corretto-11.0.23.9.1 (build 11.0.23+9-LTS, mixed mode)
```

- [Apache Maven](#)。如果你還沒有安裝阿帕奇 Maven 的。要了解如何安裝它，請參閱[安裝阿帕奇 Maven](#) 的。
  - 若要測試您的 Apache Maven 安裝，輸入以下資訊：

```
$ mvn -version
```



- IDE用於當地發展。我們建議您使用 [Eclipse Java 霓虹燈](#) 或 [IntelliJ](#) 等開發環境IDEA來開發和編譯您的應用程式。
- 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 [設定 AWS 帳戶並建立管理員使用者](#)。

## 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：

### 註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 打開<https://portal.aws.amazon.com/billing/註冊>。
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。

當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。作為安全最佳實務，請將管理存取權指派給使用者，並且僅使用根使用者來執行[需要根使用者存取權的任務](#)。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時前往 <https://aws.amazon.com/>並選擇「我的帳戶」，檢視目前的帳戶活動並管理您的帳戶。

### 建立具有管理存取權的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

## 保護您的 AWS 帳戶根使用者

1. 選擇 Root 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. 為您的 root 使用者開啟多因素驗證 (MFA)。

如需指示，請參閱《[使用指南](#)》中的「IAM為 AWS 帳戶 root 使用者啟用虛擬MFA裝置 (主控台)」。

## 建立具有管理存取權的使用者

1. 啟用IAM身分識別中心。

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身分識別中心中，將管理存取權授與使用者。

[若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。](#)

## 以具有管理存取權的使用者身分登入

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

如需使用IAM身分識別中心使用者[登入的說明](#)，請參閱使用指南中的[登入 AWS 存取入口網站](#)。AWS 登入

## 指派存取權給其他使用者

1. 在 IAM Identity Center 中，建立遵循套用最低權限權限的最佳作法的權限集。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的[建立許可集](#)。

2. 將使用者指派至群組，然後對該群組指派單一登入存取權。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的[新增群組](#)。

## 授與程式設計存取權

如果使用者想要與 AWS 之外的 AWS Management Console 授與程式設計存取 AWS 取權的方式取決於正在存取的使用者類型。

若要授與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<br><br>(在IAM身分識別中心管理的使用者) | 使用臨時認證來簽署 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。 <ul style="list-style-type: none"> <li>如需詳細資訊 AWS CLI，請參閱 <a href="#">《使 AWS CLI 用 AWS Command Line Interface 者指南》</a> AWS IAM Identity Center 中的〈配置使用〉。</li> <li>有關 AWS SDKs、工具和 AWS APIs，請參閱 <a href="#">《工具參考指南》</a> 中的 <a href="#">IAM AWS SDKs 身分識別中心驗證</a>。</li> </ul> |
| IAM                            | 使用臨時認證來簽署 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計要求。            | 請遵照 <a href="#">《使用指南》</a> 中的〈 <a href="#">搭配 AWS 源使用臨時登入資料</a> 〉中的 IAM 指示   |
| IAM                            | (不建議使用)<br>使用長期認證來簽署 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。 <ul style="list-style-type: none"> <li>對於 AWS CLI，請參閱 <a href="#">《使用者指南》</a> 中的 <a href="#">AWS Command Line Interface 〈使用 IAM 者認證進行驗證〉</a>。</li> <li>如需 AWS SDKs 和工具，請參閱和工具參考指南中的 <a href="#">使</a></li> </ul>   |

| 哪個使用者需要程式設計存取權？ | 到 | By  |
|-----------------|---|---|
|                 |   | <p><a href="#">用長期憑證進行驗證</a>。AWS SDKs</p> <ul style="list-style-type: none"><li>對於 AWS APIs，請參閱《<a href="#">使用指南</a>》中的〈<a href="#">管理使用 IAM 用 IAM 者的存取金鑰</a>〉。</li></ul> |

## 後續步驟

### [設置 AWS Command Line Interface \( AWS CLI \)](#)

## 設置 AWS Command Line Interface ( AWS CLI )

在此步驟中，您會下載並設定 AWS CLI 要與 Apache Flink 的受管理服務搭配使用。

### 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 AWS CLI 安裝了，則可能需要升級才能獲得最新功能。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](#)。若要檢查的版本 AWS CLI，請執行下列命令：

```
aws --version
```

本自學課程中的練習需要以下 AWS CLI 版本或更高版本：

```
aws-cli/1.16.63
```

## 若要設定 AWS CLI

1. 下載和設定 AWS CLI。如需相關指示，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. 在檔案中新增管理員使用者的具名設定 AWS CLI config 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，請參閱中的[區域和端點 Amazon Web Services 一般參考](#)。

### Note

本教學課程中的範例程式碼和命令使用 us-east-1 美國東部 (維吉尼亞北部) 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。

## 下一步驟

### [為 Apache Flink 應用程式建立並執行受管理的服務](#)

## 為 Apache Flink 應用程式建立並執行受管理的服務

在此步驟中，您會建立 Apache Flink 應用程式的受管服務，並將 Kinesis 資料串流當做來源和接收器。

本節包含下列步驟：

- [建立相依資源](#)
- [設定您的本機開發環境](#)
- [下載並檢查阿帕奇 Flink 流 Java 代碼](#)
- [將樣本記錄寫入輸入流](#)
- [在本機執行應用程式](#)
- [觀察 Kinesis 串流中的輸入和輸出資料](#)
- [停止應用程式在本機執行](#)
- [編譯並封裝應用程式程式碼](#)
- [上傳應用程式代碼JAR文件](#)
- [建立並設定 Apache Flink 應用程式的受管理服務](#)
- [下一步驟](#)

## 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：

- 用於輸入和輸出的兩個 Kinesis 資料串流
- 用於存放應用程式程式碼的 Amazon S3 儲存貯體

### Note

本教學課程假設您正在 us-east-1 美國東部 (維吉尼亞北部) 區域部署應用程式。如果您使用其他區域，請相應地調整所有步驟。

## 建立兩個 Amazon Kinesis 資料串流

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 主控台或下列 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](#)。若要使用建立串流 AWS CLI，請使用下列指令，調整至您用於應用程式的 [區域]。

## 建立資料串流 (AWS CLI)

1. 若要建立第一個串流 (ExampleInputStream)，請使用下列 Amazon Kinesis create-stream AWS CLI 命令：

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

```

2. 若要建立應用程式用來寫入輸出的第二個資料流，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-east-1 \  

```

## 為應用程式程式碼建立 Amazon S3 儲存貯體

您可以使用主控台建立 Amazon S3 儲存貯體。若要了解如何使用主控台建立 Amazon S3 儲存貯體，請參閱 [Amazon S3 使用者指南](#) 中的 [建立](#) 儲存貯體。使用全域唯一名稱來命名 Amazon S3 儲存貯體，例如透過附加您的登入名稱。

### Note

請務必在您用於本教學課程的「地區」中建立值區 (us-east-1)。

## 其他資源

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

- 名為 /AWS/KinesisAnalytics-java/<my-application> 的日誌群組。
- 名為 kinesis-analytics-log-stream 的日誌串流。

## 設定您的本機開發環境

對於開發和除錯，您可以直接從您選擇IDE的電腦上執行 Apache Flink 應用程式。任何阿帕奇 Flink 依賴關係都像使用阿帕奇 Maven 的常規 Java 依賴關係處理。

### Note

在你的開發機器上，你必須安裝 Java JDK 11，Maven 和 Git。我們建議您使用開發環境，如[日食 Java 霓虹燈](#)或 [IntelliJ IDEA](#)。若要確認您符合所有先決條件，請參閱[滿足完成練習的先決條件](#)。您不需要在您的機器上安裝 Apache Flink 叢集。

## 驗證您的 AWS 會話

應用程式會使用 Kinesis 資料串流來發佈資料。在本機執行時，您必須擁有具有寫入 Kinesis 資料串流之權限的有效 AWS 驗證工作階段。請使用下列步驟來驗證您的工作階段：

1. 如果您沒有設定 AWS CLI 具有有效認證的具名設定檔，請參閱[設置 AWS Command Line Interface \(AWS CLI\)](#)。
2. 發佈下列測試記錄，確認您的設定正確，且您的使用者具有寫入 Kinesis 資料串流的權限：AWS CLI

```
$ aws kinesis put-record --stream-name ExampleOutputStream --data TEST --partition-key TEST
```

3. 如果您IDE有要整合的外掛程式 AWS，您可以使用該外掛程式將憑證傳遞至在IDE。如需詳細資訊，請參閱[IntelliJ 的AWS 工具](#)[AWS組IDEA](#)和 [Eclipse 的工具組](#)。

## 下載並檢查阿帕奇 Flink 流 Java 代碼

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-managed-service-for-apache-flink-examples.git
```

2. 導覽至 `amazon-managed-service-for-apache-flink-examples/tree/main/java/GettingStarted` 目錄。



## 檢閱應用程式元

該應用程式完全在 `com.amazonaws.services.msf.BasicStreamingJob` 類中實現。此 `main()` 法會定義處理串流資料並執行資料流程的資料流程。

### Note

為了獲得最佳化的開發人員體驗，該應用程式的設計目的是在適用於 Apache Flink 的 Amazon 受管服務和本機上執行任何程式碼變更，以便在您的 IDE 中。

- 若要讀取執行階段組態，以便在 Apache Flink 的 Amazon 受管服務中執行時運作 IDE，應用程式會自動偵測它是否在 IDE。在這種情況下，應用程式以不同的方式加載運行時配置：
  - 當應用程式檢測到它在您的獨立模式下運行時 IDE，形成包含在項目資源 `application_properties.json` 文件夾中的文件。檔案的內容如下。
  - 當應用程式在適用於 Apache Flink 的 Amazon 受管服務中執行時，預設行為會從您將在 Apache Flink 應用程式的 Amazon 受管服務中定義的執行時期屬性載入應用程式組態。請參閱 [建立並設定 Apache Flink 應用程式的受管理服務](#)。

```
private static Map<String, Properties>
loadApplicationProperties(StreamExecutionEnvironment env) throws IOException {
    if (env instanceof LocalStreamEnvironment) {
        LOGGER.info("Loading application properties from '{}'",
LOCAL_APPLICATION_PROPERTIES_RESOURCE);
        return KinesisAnalyticsRuntime.getApplicationProperties(
            BasicStreamingJob.class.getClassLoader()

.getResource(LOCAL_APPLICATION_PROPERTIES_RESOURCE).getPath());
    } else {
        LOGGER.info("Loading application properties from Amazon Managed Service for
Apache Flink");
        return KinesisAnalyticsRuntime.getApplicationProperties();
    }
}
```

- 該 `main()` 方法定義應用程式數據流並運行它。
  - 初始化預設串流環境。在此範例中，我們將示範如何同時建立 `StreamExecutionEnvironment` 要與 `DataStream API` 和搭配使

用 `StreamTableEnvironment` 的和 `Table` 一起 SQL 使用的 API。這兩個環境對象是對同一運行時環境的兩個單獨的引用，以使用不同的 APIs。

```
StreamExecutionEnvironment env =  
    StreamExecutionEnvironment.getExecutionEnvironment();
```

- 載入應用程式組態參數。這將自動從正確的位置加載它們，具體取決於應用程式運行的位置：

```
Map<String, Properties> applicationParameters = loadApplicationProperties(env);
```

- 應用程式使用 [Kinesis 取用者](#) 連接器定義來源，從輸入串流讀取資料。輸入流的配置在 `PropertyGroupId = 中定義InputStream0`。串流的名稱和區域 `aws.region` 分別位於命名 `stream.name` 的屬性中。為了簡單起見，此源將記錄讀取為字符串。

```
private static FlinkKinesisConsumer<String> createSource(Properties  
    inputProperties) {  
    String inputStreamName = inputProperties.getProperty("stream.name");  
    return new FlinkKinesisConsumer<>(inputStreamName, new SimpleStringSchema(),  
        inputProperties);  
}  
...  
  
public static void main(String[] args) throws Exception {  
    ...  
    SourceFunction<String> source =  
        createSource(applicationParameters.get("InputStream0"));  
    DataStream<String> input = env.addSource(source, "Kinesis Source");  
    ...  
}
```

- 接著，應用程式會使用 [Kinesis Streams 接收器](#) 定義接收器，將資料傳送至輸出串流。輸出流名稱和區域在 `PropertyGroupId = 中定義OutputStream0`，類似於輸入流。接收器直接連接到從源獲取數據的內部 `DataStream`。在真實的應用程式中，您在源和接收器之間進行了一些轉換。

```
private static KinesisStreamsSink<String> createSink(Properties outputProperties) {  
    String outputStreamName = outputProperties.getProperty("stream.name");  
    return KinesisStreamsSink.<String>builder()  
        .setKinesisClientProperties(outputProperties)  
        .setSerializationSchema(new SimpleStringSchema())  
        .setStreamName(outputStreamName)  
        .setPartitionKeyGenerator(element ->  
            String.valueOf(element.hashCode()))
```

```
        .build();
    }
    ...
    public static void main(String[] args) throws Exception {
        ...
        Sink<String> sink = createSink(applicationParameters.get("OutputStream0"));
        input.sinkTo(sink);
        ...
    }
}
```

- 最後，您執行剛才定義的資料流程。這必須是該main()方法的最後一條指令，在定義了數據流需要的所有運算符之後：

```
env.execute("Flink streaming Java API skeleton");
```

## 使用 pom.xml 檔案

pom.xml 文件定義了應用程序所需的所有依賴關係，並設置了 Maven Shade 插件來構建包含 Flink 所需的所有依賴關係的脂肪罐。

- 一些依賴關係具有provided範圍。當應用程式在適用於 Apache Flink 的 Amazon 受管服務中執行時，這些相依性會自動提供。他們需要編譯應用程式，或在IDE。如需詳細資訊，請參閱[在本機執行應用程式](#)。請確定您使用的 Flink 版本與您將在 Apache Flink 的 Amazon 受管服務中使用的執行階段相同。

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-clients</artifactId>
  <version>${flink.version}</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-streaming-java</artifactId>
  <version>${flink.version}</version>
  <scope>provided</scope>
</dependency>
```

- 您必須將其他 Apache Flink 相依性新增至具有預設範圍的 pom，例如此應用程式使用的 [Kinesis 連接器](#)。如需詳細資訊，請參閱[搭配 Apache Flink 的受管理服務使用 Apache Flink 連接器](#)。您也可以新增應用程式所需的任何其他 Java 相依性。

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-kinesis</artifactId>
  <version>${aws.connector.version}</version>
</dependency>
```

- Maven Java 編譯器插件可以確保代碼是根據 Java 11 編譯的，這是阿帕奇 Flink 目前支持的JDK版本。
- Maven Shade 插件包裝脂肪罐，不包括運行時提供的一些庫。它還指定了兩個變壓器：ServicesResourceTransformer和ManifestResourceTransformer。後者配置包含啟動應用程式的main方法的類。如果重命名主類，請不要忘記更新此變壓器。

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-shade-plugin</artifactId>
  ...
  <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
    <mainClass>com.amazonaws.services.msf.BasicStreamingJob</mainClass>
  </transformer>
  ...
</plugin>
```

## 將樣本記錄寫入輸入流

在本節中，您會將範例記錄傳送至串流以供應用程式處理。您有兩個選項可以使用 Python 指令碼或 [Kinesis 資料產生器](#)來產生範例資料。

### 使用 Python 腳本生成示例數據

您可以使用 Python 腳本將示例記錄發送到流。

**Note**

要運行這個 Python 腳本，你必須使用 Python 3.x 和有 [Python \(博托\)](#) 庫安裝。AWS SDK

若要開始將測試資料傳送至 Kinesis 輸入串流：

1. 從數據生成器 [GitHub 存儲庫下載數據生成器 stock.py](#) Python 腳本。
2. 執行 `stock.py` 指令碼：

```
$ python stock.py
```

當您完成教學課程的其餘部分時，請保持指令碼的執行狀態。您現在可以執行您的 Apache Flink 應用程式。

## 使用 Kinesis 資料產生器產生範例資料

或者，若要使用 Python 指令碼，您也可以使用 [託管版本](#) 提供的 [Kinesis 資料產生器](#)，將隨機範例資料傳送至串流。Kinesis 資料產生器會在您的瀏覽器中執行，而且您不需要在電腦上安裝任何項目。

若要設定和執行 Kinesis 資料產生器：

1. 請遵循 [Kinesis 資料產生器文件](#) 中的指示來設定工具的存取權。您將執行設定使用者和密碼的 AWS CloudFormation 範本。
2. 透過 CloudFormation 範本產生的存取 Kinesis 資料 URL 產生器。CloudFormation 模板完成後，您可以 URL 在「輸出」選項卡中找到。
3. 設定資料產生器：
  - 區域：選取您在此教學課程中使用的區域：us-east-1
  - 串流/交付串流：選取應用程式將使用的輸入串流：ExampleInputStream
  - 每秒記錄數:100
  - 記錄範本：複製並貼上下列範本：

```
{
  "event_time" : "{{date.now("YYYY-MM-DDTkk:mm:ss.SSSSS")}}",
  "ticker" : "{{random.arrayElement(
    ["AAPL", "AMZN", "MSFT", "INTC", "TBV"]
  )}}",
```

```
"price" : {{random.number(100)}}
}
```

4. 測試範本：選擇「測試範本」，然後確認產生的記錄與下列內容類似：

```
{ "event_time" : "2024-06-12T15:08:32.04800", "ticker" : "INTC", "price" : 7 }
```

5. 啟動資料產生器：選擇「選取傳送資料」。

Kinesis 資料產生器現在正在將資料傳送至 `ExampleInputStream`

## 在本機執行應用程式

您可以在 IDE

### Note

在繼續之前，請確認輸入和輸出串流是否可用。請參閱 [建立兩個 Amazon Kinesis 資料串流](#)。另外，請確認您具有從兩個串流讀取和寫入的權限。請參閱 [驗證您的 AWS 會話](#)。設置本地開發環境需要 Java 11JDK，阿帕奇 Maven 的，並IDE用於 Java 開發。確認您符合必要的先決條件。請參閱 [滿足完成練習的先決條件](#)。

## 將 Java 專案匯入您的 IDE

若要在中開始處理應用程式IDE，您必須將其匯入為 Java 專案。

您複製的儲存庫包含多個範例。每個例子都是一個單獨的項目。在本教學課程中，請將 `./java/GettingStarted` 子目錄中的內容匯入您的IDE。

將代碼作為使用 Maven 的現有 Java 項目插入。

### Note

導入新 Java 項目的確切過程根據IDE您使用的不同而有所不同。

## 檢查本地應用程式配置

在本機執行時，應用程式會使用下專 `application.properties.json` 案資源資料夾中檔案中的組態 `./src/main/resources`。您可以編輯此檔案以使用不同的 Kinesis 串流名稱或區域。

```
[
  {
    "PropertyGroupId": "InputStream0",
    "PropertyMap": {
      "stream.name": "ExampleInputStream",
      "flink.stream.initpos": "LATEST",
      "aws.region": "us-east-1"
    }
  },
  {
    "PropertyGroupId": "OutputStream0",
    "PropertyMap": {
      "stream.name": "ExampleOutputStream",
      "aws.region": "us-east-1"
    }
  }
]
```

## 設置您的IDE運行配置

您可以透過執行主類別 `com.amazonaws.services.msf.BasicStreamingJob`，IDE 直接從您的 Flink 應用程式執行和偵錯，就像執行任何 Java 應用程式一樣。在運行應用程序之前，您必須設置運行配置。設置取決於 IDE 您正在使用的。例如，請參閱 IntelliJ IDEA J 文檔中的 [運行/調試配置](#)。特別是，您必須設定下列項目：

1. 將 **provided** 依賴關係添加到類路徑中。這是必要的，以確保在本地運行時將具有 provided 範圍的依賴關係傳遞給應用程序。如果沒有此設定，應用程式會立即顯示 `class not found` 錯誤。
2. 傳遞 AWS 認證以存取 Kinesis 串流至應用程式。最快的方法是使用 [IntelliJ IDEA 的 AWS 工具包](#)。在 Run 配置中使用此 IDE 插件，您可以選擇特定的配置 AWS 文件。AWS 使用此設定檔進行驗證。您不需要直接傳遞 AWS 憑證。
3. 驗證是否使用 JDK11 IDE 執行應用程式。

## 在您的中運行應用程序 IDE

設定的 [執行] 組態之後 `BasicStreamingJob`，您可以像執行一般 Java 應用程式一樣執行或偵錯它。

**Note**

您不能直接 `java -jar ...` 從命令行運行 Maven 生成的脂肪罐。此 jar 不包含獨立執行應用程式所需的 Flink 核心相依性。

當應用程式成功啟動時，它會記錄有關獨立迷你叢集和連接器初始化的一些資訊。後面是 Flink 通常會在應用程式啟動時發出的 INFO 一些 WARN 日誌。

```
13:43:31,405 INFO com.amazonaws.services.msf.BasicStreamingJob [] -
  Loading application properties from 'flink-application-properties-dev.json'
13:43:31,549 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Flink Kinesis Consumer is going to read the following streams:
  ExampleInputStream,
13:43:31,676 INFO org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils []
 - The configuration option taskmanager.cpu.cores required for local execution is not
 set, setting it to the maximal possible value.
13:43:31,676 INFO org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils
 [] - The configuration option taskmanager.memory.task.heap.size required for local
 execution is not set, setting it to the maximal possible value.
13:43:31,676 INFO org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils []
 - The configuration option taskmanager.memory.task.off-heap.size required for local
 execution is not set, setting it to the maximal possible value.
13:43:31,676 INFO org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils []
 - The configuration option taskmanager.memory.network.min required for local execution
 is not set, setting it to its default value 64 mb.
13:43:31,676 INFO org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils []
 - The configuration option taskmanager.memory.network.max required for local execution
 is not set, setting it to its default value 64 mb.
13:43:31,676 INFO org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils [] -
 The configuration option taskmanager.memory.managed.size required for local execution
 is not set, setting it to its default value 128 mb.
13:43:31,677 INFO org.apache.flink.runtime.minicluster.MinicCluster [] -
  Starting Flink Mini Cluster
....
```

初始化完成後，應用程式不會發出任何進一步的記錄項目。當資料流動時，不會發出任何記錄檔。

若要驗證應用程式是否正確處理資料，您可以檢查輸入和輸出 Kinesis 串流，如下節所述。



### Note

不發出有關流動資料的記錄是 Flink 應用程式的正常行為。在每個記錄上發出日誌可能很方便調試，但在生產環境中運行時可能會增加相當大的開銷。

## 觀察 Kinesis 串流中的輸入和輸出資料

您可以使用 Amazon Kinesis 主控台中的資料檢視器，觀察 (產生範例 Python) 或 Kinesis 資料產生器 (連結) 傳送至輸入串流的記錄。

### 觀察記錄

1. 在 <https://console.aws.amazon.com/Kinesis> 處開啟室壁運動主控台。
2. 確認區域與您執行此教學課程的位置相同，依預設為 us-east-1 美國東部 (維吉尼亞北部)。如果「區域」不相符，請變更該區域。
3. 選擇「資料串流」。
4. 選取您要觀察的串流，ExampleInputStream或 ExampleOutputStream。
5. 選擇「資料檢視器」標籤。
6. 選擇任何碎片，保持最新作為起始位置，然後選擇獲取記錄。您可能會看到「找不到此要求的記錄」錯誤訊息。如果是這樣，請選擇重試取得記錄。發佈至串流的最新記錄會顯示。
7. 在「資料」欄中選擇值，以JSON格式檢查記錄的內容。

## 停止應用程式在本機執行

停止在您的IDE。通IDE常會提供「停止」選項。確切的位置和方法取決於IDE您使用的。

## 編譯並封裝應用程式程式碼

在本節中，您可以使用阿帕奇 Maven 編譯 Java 代碼並將其打包到一個JAR文件中。您可以使用 Maven 命令行工具或IDE。

若要使用 Maven 命令列進行編譯和封裝：

移至包含 Java GettingStarted 專案的目錄，並執行下列命令：

```
$ mvn package
```

要使用以下命令編譯和打包IDE：

`mvn package`從你的 IDE Maven 集成運行。

在這兩種情況下，都會建立下列JAR檔案：`target/amazon-msf-java-stream-app-1.0.jar`。

#### Note

從您運行「構建項目」IDE 可能無法創建該JAR文件。

## 上傳應用程式代碼JAR文件

在本節中，您將在上一節中建立的JAR檔案上傳到您在本教學開始時建立的 Amazon 簡單儲存服務 (Amazon S3) 儲存貯體。如果您尚未完成此步驟，請參閱 ( [鏈接](#) )。

上傳應用程式代碼JAR檔

1. 在開啟 Amazon S3 主控台 <https://console.aws.amazon.com/s3/>。
2. 選擇您先前為應用程式代碼建立的值區。
3. 選擇上傳。
4. 選擇 Add files (新增檔案)。
5. 導覽至在上一個步驟中產生的JAR檔案：`target/amazon-msf-java-stream-app-1.0.jar`。
6. 選擇「上傳」而不變更任何其他設定。

#### Warning

請確定您在中選取了正確的JAR檔案`<repo-dir>/java/GettingStarted/target/amazon-msf-java-stream-app-1.0.jar`。  
該`target`目錄還包含您不需要上傳的其他JAR文件。

## 建立並設定 Apache Flink 應用程式的受管理服務

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。在本教程中，您將使用控制台。

### Note

當您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。使用建立應用程式時 AWS CLI，您需要分別建立這些資源。

## 主題

- [建立應用程式](#)
- [編輯IAM策略](#)
- [設定應用程式](#)
- [執行應用程式](#)
- [觀察正在運行的應用程序的指標](#)
- [觀察 Kinesis 串流中的輸出資料](#)
- [停止應用程式](#)

## 建立應用程式

### 建立應用程式

1. 在 <https://console.aws.amazon.com/flink> 開啟適用於阿帕奇 Flink 的受管理服務
2. 確認已選取正確的區域：us-east-1 美國東部 (維吉尼亞北部)
3. 打開右側的菜單，然後選擇 Apache Flink 應用程序，然後創建流應用程序。或者，在初始頁面的 [開始使用] 容器中選擇 [建立串流應用程式]。
4. 在 [建立串流應用程式] 頁面上：
  - 選擇設定串流處理應用程式的方法：選擇「從頭開始建立」。
  - 阿帕奇 Flink 配置，應用程序 Flink 版本：選擇阿帕奇 Flink 1.19。
5. 設定應用程式
  - 應用程式名稱：輸入 **MyApplication**。
  - 描述：輸入 **My java test app**。
  - 存取應用程式資源：選擇 [建立/更新 **kinesis-analytics-MyApplication-us-east-1** 具有必要策略的IAM角色]。

## 6. 設定應用程式設定的範本

- 範本：選擇 [開發]。

## 7. 選擇頁面底部的 [建立串流應用程式]。

### Note

當您使用主控台為 Apache Flink 應用程式建立受管理服務時，您可以選擇為應用程式建立IAM角色和原則。應用程式使用此角色和政策來存取其相依資源。這些IAM資源會使用您的應用程式名稱和區域命名，如下所示：

- 政策：`kinesis-analytics-service-MyApplication-us-east-1`
- 角色：`kinesisanalytics-MyApplication-us-east-1`

Amazon 阿帕奇 Flink 託管服務以前被稱為 Kinesis Data Analytics。自動創建的資源的名稱是 `kinesis-analytics-` 為了向後兼容性的前綴。

## 編輯IAM策略

編輯原IAM則以新增存取 Kinesis 資料串流的權限。

### 若要編輯策略

1. 在開啟IAM主控台<https://console.aws.amazon.com/iam/>。
2. 選擇政策。選擇主控台為您在上節所建立的 `kinesis-analytics-service-MyApplication-us-east-1` 政策。
3. 選擇 [編輯]，然後選擇索引標籤。
4. 將下列政策範例的反白部分新增至政策。取代範例帳戶 IDs (`012345678901`) 使用您的帳戶 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
```

```

        "s3:GetObjectVersion"
    ],
    "Resource": [
        "arn:aws:s3:::my-bucket/kinesis-analytics-placeholder-s3-object"
    ]
},
{
    "Sid": "ListCloudwatchLogGroups",
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogGroups"
    ],
    "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:*"
    ]
},
{
    "Sid": "ListCloudwatchLogStreams",
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogStreams"
    ],
    "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
    ]
},
{
    "Sid": "PutCloudwatchLogs",
    "Effect": "Allow",
    "Action": [
        "logs:PutLogEvents"
    ],
    "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
},
{
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleInputStream"
}

```

```

    },
    {
      "Sid": "WriteOutputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleOutputStream"
    }
  ]
}

```

5. 選擇頁面底部的 [下一步]，然後選擇 [儲存變更]。

## 設定應用程式

編輯應用程式組態以設定應用程式程式碼人工因素。

### 編輯模型組態

1. 在MyApplication頁面上，選擇設定。
2. 在應用程式程式碼位置區段中：
  - 對於 Amazon S3 儲存貯體，請選取先前為應用程式程式碼建立的儲存貯體。選擇 [瀏覽] 並選取正確的值區，然後選取 [選擇]。請勿按一下值區名稱。
  - 對於 Amazon S3 物件的路徑，請輸入 **amazon-msf-java-stream-app-1.0.jar**。
3. 對於 [存取權限]，選擇 [建立/更新kinesis-analytics-MyApplication-us-east-1具有必要策略的IAM角色]
4. 在 [執行階段屬性] 區段中，新增下列屬性。
5. 選擇「新增項目」，然後新增下列每個參數：

| 群組 ID                | 金鑰                 | 值                          |
|----------------------|--------------------|----------------------------|
| <b>InputStream0</b>  | <b>stream.name</b> | <b>ExampleInputStream</b>  |
| <b>InputStream0</b>  | <b>aws.region</b>  | <b>us-east-1</b>           |
| <b>OutputStream0</b> | <b>stream.name</b> | <b>ExampleOutputStream</b> |
| <b>OutputStream0</b> | <b>aws.region</b>  | <b>us-east-1</b>           |

6. 請勿修改任何其他區段。
7. 選擇 Save changes (儲存變更)。

### 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

該應用程式現在已配置並準備好運行。

### 執行應用程式

1. 在適用於 Apache Flink 的 Amazon 受管服務的主控台上，選擇我的應用程式，然後選擇執行。
2. 在下一頁的 [應用程式還原組態] 頁面上，選擇 [以最新快照執行]，然後選擇 [執行]。

「應用程式中的狀態」詳細資訊會從Ready應用程式啟動到再轉換為應用程式的Running時間。Starting

當應用程式處於狀Running態時，您現在可以開啟 Flink 儀表板。

### 開啟 儀表板

1. 選擇「開啟阿帕奇 Flink 儀表板」。儀表板會在新頁面上開啟。
2. 在「執行工作」清單中，選擇您可以看到的單一工作。

### Note

如果您設定執行階段屬性或編輯不正確的IAM原則，應用程式狀態可能會變成Running，但 Flink 儀表板會顯示工作正在持續重新啟動。如果應用程式設定錯誤或缺少存取外部資源的權限，這是常見的失敗案例。

發生這種情況時，請檢查 Flink 儀表板中的「例外」標籤以查看問題的原因。

## 觀察正在運行的應用程式的指標

在MyApplication頁面的 Amazon CloudWatch 指標區段中，您可以看到執行中應用程式中的一些基本指標。

若要檢視測量結果

1. 在「重新整理」按鈕旁邊，從下拉式清單中選取 10 秒。
2. 當應用程式執行且健康狀態良好時，您可以看到正常執行時間指標不斷增加。
3. 完整重新啟動指標應為零。如果正在增加，則配置可能有問題。若要調查問題，請檢閱 Flink 儀表板上的「例外」標籤。
4. 在運作狀況良好的應用程式中，失敗的檢查點數量應為零。

### Note

此儀表板會顯示一組固定的指標，精細度為 5 分鐘。您可以使用儀表板中的任何指標來建立自訂應用程式 CloudWatch 儀表板。

## 觀察 Kinesis 串流中的輸出資料

請確定您仍在使用 Python 指令碼或 Kinesis 資料產生器將資料發佈至輸入。

您現在可以使用中的「資料檢視器」，觀察在 Apache Flink 受管理服務上執行之應用程式的輸出 <https://console.aws.amazon.com/kinesis/>，與您之前已執行的操作類似。

若要檢視輸出

1. 在 <https://console.aws.amazon.com/Kinesis> 處開啟室壁運動主控台。
2. 確認區域與您用來執行此自學課程的區域相同。依預設，它是美國東部-1 美國東部 (維吉尼亞北部)。如有必要，請變更「區域」。
3. 選擇「資料串流」。
4. 選取您要觀察的串流。在本教學課程中，使用 ExampleOutputStream。
5. 選擇「資料檢視器」標籤。
6. 選取任何 [碎片]，將 [最新] 保留為起始位置，然後選擇 [取得記錄]。您可能會看到「找不到此要求的記錄」錯誤訊息。如果是這樣，請選擇重試取得記錄。發佈至串流的最新記錄會顯示。
7. 在「資料」欄中選取值，以JSON格式檢查記錄的內容。



## 停止應用程式

若要停止應用程式，請移至名為的 Apache Flink 應用程式的受管理服務的主控台頁面。MyApplication

### 停止應用程式

1. 從「動作」下拉式清單中，選擇「停止」。
2. 「應用程式中的狀態」詳細資訊Ready會從應Running用程式完全停止轉換為Stopping，然後再轉換為應用程式的

#### Note

別忘了也停止將資料從 Python 指令碼或 Kinesis 資料產生器傳送至輸入串流。

## 下一步驟

### [清理 AWS 資源](#)

## 清理 AWS 資源

本節包括清理在此入門 (DataStream API) 教學課程中建立的 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的IAM資源](#)
- [刪除您的 CloudWatch 資源](#)
- [探索阿帕奇 Flink 的其他資源](#)

## 刪除適用於 Apache Flink 應用程式的受管理服務

請使用下列程序刪除應用程式。

1. 在 <https://console.aws.amazon.com/Kinesis> 處開啟室壁運動主控台。

2. 在適用於 Apache Flink 的受管理服務面板中，選擇 MyApplication。
3. 從「動作」下拉式清單中，選擇「刪除」，然後確認刪除。

## 刪除您的 Kinesis 資料串流

1. 在 <https://console.aws.amazon.com/flink> 開啟適用於 Apache Flink 的受管理服務主控台。
2. 選擇 [資料串流]。
3. 選取您建立的兩個串流，ExampleInputStream 和 ExampleOutputStream。
4. 從「動作」下拉式清單中，選擇「刪除」，然後確認刪除。

## 刪除您的 Amazon S3 對象和儲存桶

使用下列程序刪除您的 Amazon S3 物件和儲存貯體。

若要從 S3 儲存貯體刪除物件

1. 在開啟 Amazon S3 主控台 <https://console.aws.amazon.com/s3/>。
2. 選取您為應用程式成品建立的 S3 儲存貯體。
3. 選取您上傳的應用程式人工因素，命名為 amazon-msf-java-stream-app-1.0.jar。
4. 選擇刪除並確認刪除。

刪除 S3 儲存貯體

1. 在開啟 Amazon S3 主控台 <https://console.aws.amazon.com/s3/>。
2. 選取您為成品建立的值區。
3. 選擇刪除並確認刪除。

### Note

S3 儲存貯體必須是空的才能刪除它。

## 刪除您的IAM資源

若要刪除您的IAM資源

1. 在開啟IAM主控台<https://console.aws.amazon.com/iam/>。
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-ea st-1 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 us-east- 1 角色。
8. 選擇刪除角色，然後確認刪除。

## 刪除您的 CloudWatch 資源

1. 在開啟 CloudWatch 主控台<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，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。

## 探索阿帕奇 Flink 的其他資源

[探索其他資源](#)

### 探索其他資源

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。

- [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計程車公司的運營。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。
- [為 Apache Flink 應用程式建立及使用受管理服務的範例](#)：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充的串流ETL、分析和事件驅動型應用程式。

## 開始使用阿帕奇 Flink 的 Amazon 託管服務 (表API)

本節將為您介紹 Apache Flink 的受管理服務的基本概念，以及使用資料表API和SQL。它示範如何在相同應用程式APIs中切換不同的應用程式，並說明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

- [檢閱 Apache Flink 應用程式的受管理服務元件](#)
- [完成必要的先決條件](#)
- [為 Apache Flink 應用程式建立並執行受管理的服務](#)
- [下一步驟](#)
- [清理 AWS 資源](#)
- [探索其他資源](#)

## 檢閱 Apache Flink 應用程式的受管理服務元件

### Note

Apache Flink 的託管服務支持所有 [Apache Flink APIs](#) 和潛在的所有JVM語言。視API您選擇的項目而定，應用程式的結構和實作會稍有不同。本教程介紹了使用表的應用程序的實現 SQL，API並與在 Java 中實現的集成。DataStream API

為了處理資料，Apache Flink 應用程式的受管理服務會使用 Java 應用程式，該應用程式會使用 Apache Flink 執行階段處理輸入並產生輸出。

一個典型的阿帕奇 Flink 應用程序具有以下組件：

- 執行階段屬性：您可以使用執行階段屬性，將組態參數傳遞至應用程式，而不必修改和重新發佈程式碼。
- 來源：應用程序消耗來自一個或多個來源的數據。來源使用[連接器](#)從外部系統讀取資料，例如 Kinesis 資料串流或 Amazon MSK 主題。對於開發或測試，你也可以有隨機源 [ly 生成測試數據。如需詳細資訊，請參閱[將串流資料來源新增至 Apache Flink 的受管理服務](#)。使用SQL或表格API，來源定義為來源表格。

- **轉換**：應用程式會透過一或多個轉換來處理資料，以篩選、豐富或彙總資料。使用SQL或表格時API，轉換被定義為對資料表或檢視表的查詢。
- **接收器**：應用程式通過接收器將數據發送到外部系統。接收器使用[連接器](#)將資料傳送到外部系統，例如 Kinesis 資料串流、Amazon MSK 主題、Amazon S3 儲存貯體或關聯式資料庫。您也可以使用特殊的連接器列印輸出，僅用於開發目的。使用SQL或表格時API，接收器會定義為接收器表格，您將在其中插入結果。如需詳細資訊，請參閱[在 Apache Flink 的受管理服務中使用接收器寫入資料](#)。

您的應用程式需要一些外部相依性，例如應用程式使用的 Flink 連接器，或可能是 Java 程式庫。若要在適用於 Apache Flink 的 Amazon 受管服務中執行，您必須將應用程式與相依性一起封裝到胖子中，JAR然後將其上傳到 Amazon S3 儲存貯體。然後建立 Managed Service for Apache Flink 應用程式。您可以傳遞程式碼套件位置，以及其他執行階段設定參數。本教學課程將示範如何使用 Apache Maven 封裝應用程式，以及如何在您選擇IDE的項目中在本機執行應用程式。

## 完成必要的先決條件

開始本教學課程之前，請先完成 [開始使用阿帕奇 Flink 的 Amazon 託管服務 \(\) DataStream API](#) 中的前兩個步驟：

- [滿足完成練習的先決條件](#)
- [設置 AWS Command Line Interface \( AWS CLI \)](#)

若要開始使用，請參閱 [建立應用程式](#)。

## 為 Apache Flink 應用程式建立並執行受管理的服務

在本練習中，您會建立 Apache Flink 應用程式的受管理服務，並將 Kinesis 資料串流當做來源和接收器。

本節包含下列步驟：

- [建立相依資源](#)
- [設定您的本機開發環境](#)
- [下載並檢查阿帕奇 Flink 流 Java 代碼](#)
- [在本機執行應用程式](#)
- [觀察將資料寫入 S3 儲存貯體的應用程式](#)
- [停止應用程式在本機執行](#)

- [編譯並封裝應用程式程式碼](#)
- [上傳應用程式碼JAR檔案](#)
- [建立並設定 Apache Flink 應用程式的受管理服務](#)

## 建立相依資源

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

- 用於存放應用程式程式碼和編寫應用程式輸出的 Amazon S3 儲存貯體。

### Note

本教學課程假設您正在 us-east-1 區域中部署應用程式。如果您使用其他區域，則必須相應地調整所有步驟。

## 建立 Amazon S3 儲存貯體

您可以使用主控台建立 Amazon S3 儲存貯體。如需建立這些資源的相關指示，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](#)。透過附加您的登入名稱，為 Amazon S3 儲存貯體提供全域唯一的名稱。

### Note

請務必在您用於本教學課程的「區域」中建立值區。本教學課程的預設值為 us-east-1。

## 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/<my-application> 的日誌群組。
- 名為 kinesis-analytics-log-stream 的日誌串流。

## 設定您的本機開發環境

對於開發和除錯，您可以直接從您選擇IDE的電腦上執行 Apache Flink 應用程式。任何阿帕奇 Flink 依賴關係被處理為使用 Maven 的正常 Java 依賴關係。

### Note

在你的開發機器上，你必須安裝 Java JDK 11，Maven 和 Git。我們建議您使用開發環境，如[日食 Java 霓虹燈](#)或[IntelliJ IDEA](#)。若要確認您符合所有先決條件，請參閱[滿足完成練習的先決條件](#)。您不需要在您的機器上安裝 Apache Flink 叢集。

## 驗證您的 AWS 會話

應用程式會使用 Kinesis 資料串流來發佈資料。在本機執行時，您必須擁有具有寫入 Kinesis 資料串流之權限的有效 AWS 驗證工作階段。請使用下列步驟來驗證您的工作階段：

1. 如果您沒有設定 AWS CLI 有效認證的具名設定檔，請參閱[設置 AWS Command Line Interface \(AWS CLI\)](#)。
2. 如果您IDE有要整合的外掛程式 AWS，您可以使用該外掛程式將憑證傳遞至在IDE。如需詳細資訊，請參閱[IntelliJ 的AWS 工具AWS組IDEA和用於編譯應用程式或執行 Eclipse 的工具組](#)。

## 下載並檢查阿帕奇 Flink 流 Java 代碼

此範例的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-managed-service-for-apache-flink-examples.git
```

2. 導覽至 `./java/GettingStartedTable` 目錄。

## 檢閱應用程式元

該應用程式完全在 `com.amazonaws.services.msf.BasicTableJob` 類中實現。該 `main()` 方法定義源，轉換和接收器。執行由此方法結尾的執行陳述式啟動。



**Note**

為了獲得最佳的開發人員體驗，該應用程式的設計目的是在適用於 Apache Flink 的 Amazon 受管服務和本機上執行任何程式碼變更，以便在您的 IDE 中。

- 若要讀取執行階段組態，以便在 Apache Flink 的 Amazon 受管服務中執行時運作 IDE，應用程式會自動偵測它是否在 IDE。在這種情況下，應用程序以不同的方式加載運行時配置：
  1. 當應用程序檢測到它在您的獨立模式下運行時 IDE，形成包含在項目資源 `application_properties.json` 文件夾中的文件。檔案的內容如下。
  2. 當應用程式在適用於 Apache Flink 的 Amazon 受管服務中執行時，預設行為會從您將在 Apache Flink 應用程式的 Amazon 受管服務中定義的執行時期屬性載入應用程式組態。請參閱 [建立並設定 Apache Flink 應用程式的受管理服務](#)。

```
private static Map<String, Properties>
loadApplicationProperties(StreamExecutionEnvironment env) throws IOException {
    if (env instanceof LocalStreamEnvironment) {
        LOGGER.info("Loading application properties from '{}'",
LOCAL_APPLICATION_PROPERTIES_RESOURCE);
        return KinesisAnalyticsRuntime.getApplicationProperties(
            BasicStreamingJob.class.getClassLoader());
    } else {
        LOGGER.info("Loading application properties from Amazon Managed Service for
Apache Flink");
        return KinesisAnalyticsRuntime.getApplicationProperties();
    }
}
```

- 該 `main()` 方法定義應用程序數據流並運行它。
- 初始化預設串流環境。在此範例中，我們將示範如何同時建立與使用 `DataStream API`、和 `Table` 搭配 SQL 使用的 API。`StreamExecutionEnvironment` `StreamTableEnvironment` 這兩個環境對象是對同一運行時環境的兩個單獨的引用，以使用不同的 APIs。

```
StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
```

```
StreamTableEnvironment tableEnv = StreamTableEnvironment.create(env,
    EnvironmentSettings.newInstance().build());
```

- 載入應用程式組態參數。這將自動從正確的位置加載它們，具體取決於應用程式運行的位置：

```
Map<String, Properties> applicationParameters = loadApplicationProperties(env);
```

- 當 Flink 完成[檢查點](#)時，應用程式用來將結果寫入 Amazon S3 輸出檔案的[接FileSystem 收器](#)連接器。您必須啟用檢查點才能將檔案寫入目的地。當應用程式在適用於 Apache Flink 的 Amazon 受管服務中執行時，應用程式組態會控制檢查點並預設啟用該檢查點。相反地，在本機執行時，預設會停用檢查點。該應用程式檢測到它在本地運行，並配置檢查點每 5,000 毫秒。

```
if (env instanceof LocalStreamEnvironment) {
    env.enableCheckpointing(5000);
}
```

- 此應用程式不會接收來自實際外部來源的資料。它會產生隨機資料以透過[DataGen 連接器](#)進行處理。此連接器可用於 DataStream API、SQL、和表格API。為了演示之間的集成 APIs，應用程式使用該 DataStream API 版本，因為它提供了更多的靈活性。每條記錄都是由 StockPriceGeneratorFunction 在這種情況下調用的生成器函數生成的，您可以在其中放置自定義邏輯。

```
DataGeneratorSource<StockPrice> source = new DataGeneratorSource<>(
    new StockPriceGeneratorFunction(),
    Long.MAX_VALUE,
    RateLimiterStrategy.perSecond(recordPerSecond),
    TypeInformation.of(StockPrice.class));
```

- 在中 DataStream API，記錄可以具有自訂類別。類必須遵循特定規則，以便 Flink 可以將其用作記錄。如需詳細資訊，請參閱[支援的資料類型](#)。在此範例中，StockPrice 類別為 [POJO](#)。
- 然後將來源附加至執行環境，產生 DataStream 的 StockPrice。此應用程式不使用[事件時語義](#)，也不會生成水印。以平行處理原則為 1 執行 DataGenerator 來源，而不受應用程式其餘部分的平行處理原則影響。

```
DataStream<StockPrice> stockPrices = env.fromSource(
    source,
    WatermarkStrategy.noWatermarks(),
    "data-generator"
).setParallelism(1);
```

- 在數據處理流程中以下內容是使用表API和定義SQL。要做到這一點，我們將 `DataStream` 的 `StockPrices` 轉換為一個表。資料表的結構定義會自動從 `StockPrice` 類別推斷出來。

```
Table stockPricesTable = tableEnv.fromDataStream(stockPrices);
```

- 下列程式碼片段會示範如何使用程式設計資料表API來定義檢視和查詢：

```
Table filteredStockPricesTable = stockPricesTable.  
    select(  
        $("eventTime").as("event_time"),  
        $("ticker"),  
        $("price"),  
        dateFormat($("eventTime"), "yyyy-MM-dd").as("dt"),  
        dateFormat($("eventTime"), "HH").as("hr")  
    ).where($("price").isGreater(50));  
  
tableEnv.createTemporaryView("filtered_stock_prices", filteredStockPricesTable);
```

- 接收器表的定義是將結果以檔案形式寫入 Amazon S3 儲JSON存貯體。為了說明以編程方式定義視圖的區別，使用 Table API 定義接收器表使用SQL。

```
tableEnv.executeSql("CREATE TABLE s3_sink (" +  
    "eventTime TIMESTAMP(3)," +  
    "ticker STRING," +  
    "price DOUBLE," +  
    "dt STRING," +  
    "hr STRING" +  
    ") PARTITIONED BY ( dt, hr ) WITH (" +  
    "'connector' = 'filesystem'," +  
    "'format' = 'json'," +  
    "'path' = 's3a://' + s3Path + "'" +  
    ")");
```

- 的最後一個步驟是將過濾後 `executeInsert()` 的股票價格視圖插入接收表中。此方法啟動到目前為止，我們已經定義的數據流的執行。

```
filteredStockPricesTable.executeInsert("s3_sink");
```

## 使用 pom.xml 檔案

pom.xml 文件定義了應用程序所需的所有依賴關係，並設置了 Maven Shade 插件來構建包含 Flink 所需的所有依賴關係的脂肪罐。

- 一些依賴關係具有 provided 範圍。當應用程式在適用於 Apache Flink 的 Amazon 受管服務中執行時，這些相依性會自動提供。他們是必需的應用程序或應用程序在您的 IDE。如需詳細資訊，請參閱 (更新至表格 API) [在本機執行應用程式](#)。請確定您使用的 Flink 版本與您將在 Apache Flink 的 Amazon 受管服務中使用的執行階段相同。若要使用表格 API 和 SQL，您必須包含 flink-table-planner-loader 和 flink-table-runtime-dependencies，兩者都有 provided 範圍。

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-streaming-java</artifactId>
  <version>${flink.version}</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-clients</artifactId>
  <version>${flink.version}</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-table-planner-loader</artifactId>
  <version>${flink.version}</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-table-runtime</artifactId>
  <version>${flink.version}</version>
  <scope>provided</scope>
</dependency>
```

- 您必須將其他 Apache Flink 依賴項添加到具有默認範圍的 pom 中。例如，[DataGen 連接器](#)、[File System SQL 連接器](#) 和 [格 JSON 式](#)。

```
<dependency>
  <groupId>org.apache.flink</groupId>
```

```

    <artifactId>flink-connector-datagen</artifactId>
    <version>${flink.version}</version>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-connector-files</artifactId>
    <version>${flink.version}</version>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-json</artifactId>
    <version>${flink.version}</version>
</dependency>

```

- 若要在本機執行時寫入 Amazon S3，S3 Hadoop 檔案系統也包含智慧provided範圍。

```

<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-s3-fs-hadoop</artifactId>
    <version>${flink.version}</version>
    <scope>provided</scope>
</dependency>

```

- Maven Java 編譯器插件可以確保代碼是根據 Java 11 編譯的，這是阿帕奇 Flink 目前支持的JDK版本。
- Maven Shade 插件包裝脂肪罐，不包括運行時提供的一些庫。它還指定了兩個變壓器：ServicesResourceTransformer和ManifestResourceTransformer。後者配置包含啟動應用程序的main方法的類。如果重命名主類，請不要忘記更新此變壓器。

```

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-shade-plugin</artifactId>
    ...
    <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
        <mainClass>com.amazonaws.services.msf.BasicStreamingJob</mainClass>
    </transformer>
    ...
</plugin>

```

## 在本機執行應用程式

您可以在 IDE

### Note

在繼續之前，請確認輸入和輸出串流是否可用。請參閱 [建立兩個 Amazon Kinesis 資料串流](#)。另外，請確認您具有從兩個串流讀取和寫入的權限。請參閱 [驗證您的 AWS 會話](#)。設置本地開發環境需要 Java 11JDK，阿帕奇 Maven 和一個IDE用於 Java 開發。確認您符合必要的先決條件。請參閱 [滿足完成練習的先決條件](#)。

## 將 Java 專案匯入您的 IDE

若要在中開始處理應用程式IDE，您必須將其匯入為 Java 專案。

您複製的儲存庫包含多個範例。每個例子都是一個單獨的項目。在本教學課程中，請將 ./jave/GettingStartedTable 子目錄中的內容匯入您的IDE。

將代碼作為使用 Maven 的現有 Java 項目插入。

### Note

導入新 Java 項目的確切過程根據IDE您使用的不同而有所不同。

## 修改本地應用程式配置

在本機執行時，應用程式會使用下專 application\_properties.json 案資源資料夾中檔案中的組態 ./src/main/resources。在此教學課程應用程式中，組態參數是值區的名稱以及將寫入資料的路徑。

編輯組態並修改 Amazon S3 儲存貯體的名稱，以符合您在本教學開始時建立的儲存貯體。

```
[
  {
    "PropertyGroupId": "bucket",
    "PropertyMap": {
      "name": "<bucket-name>",
```

```
"path": "output"  
}  
}  
]
```

### Note

例如，組態屬性name只能包含值區名稱my-bucket-name。請勿包含任何前置詞，例如s3://或結尾斜線。  
如果修改路徑，請省略任何前導或結尾的斜線。

## 設置您的IDE運行配置

您可以透過執行主類別`com.amazonaws.services.msf.BasicTableJob`，IDE直接從您的 Flink 應用程式執行和偵錯，就像執行任何 Java 應用程式一樣。在運行應用程序之前，您必須設置運行配置。該設置取決於您IDE正在使用的。例如，請參閱 IntelliJ IDEA J 文檔中的[運行/調試配置](#)。特別是，您必須設定下列項目：

1. 將**provided**依賴關係添加到類路徑中。這是必要的，以確保在本地運行時將具有provided範圍的依賴關係傳遞給應用程序。如果沒有此設定，應用程式會立即顯示class not found錯誤。
2. 傳遞 AWS 認證以存取 Kinesis 串流至應用程式。最快的方法是使用 [IntelliJ IDEA 的AWS 工具包](#)。在 Run 配置中使用此IDE插件，您可以選擇特定的配置 AWS 文件。AWS 使用此設定檔進行驗證。您不需要直接傳遞 AWS 憑證。
3. 驗證是否使用 JDK11 IDE 執行應用程式。

## 在您的中運行應用程序 IDE

設定的 [執行] 組態之後BasicTableJob，您可以像執行一般 Java 應用程式一樣執行或偵錯它。

### Note

您不能直接`java -jar ...`從命令行運行 Maven 生成的胖罐。此 jar 不包含獨立執行應用程式所需的 Flink 核心相依性。

當應用程式成功啟動時，它會記錄有關獨立迷你叢集和連接器初始化的一些資訊。後面是 Flink 通常會在應用程序啟動時發出的INFO一些WARN日誌。

```
21:28:34,982 INFO com.amazonaws.services.msf.BasicTableJob
    [] - Loading application properties from 'flink-application-properties-
dev.json'
21:28:35,149 INFO com.amazonaws.services.msf.BasicTableJob
    [] - s3Path is ExampleBucket/my-output-bucket
...
```

初始化完成後，應用程式不會發出任何進一步的記錄項目。當資料流動時，不會發出任何記錄檔。

若要驗證應用程式是否正確處理資料，您可以檢查輸出值區的內容，如下節所述。

### Note

不發出有關流動資料的記錄是 Flink 應用程式的正常行為。在每個記錄上發出日誌可能很方便調試，但在生產環境中運行時可能會增加相當大的開銷。

## 觀察將資料寫入 S3 儲存貯體的應用程式

此範例應用程式會在內部產生隨機資料，並將此資料寫入您設定的目標 S3 儲存貯體。除非您修改了預設組態路徑，否則資料將以格式寫入 output 路徑，後面接著資料和小時分區 `./output/<yyyy-MM-dd>/<HH>`。

[接 FileSystem 收器連接器](#)會在 Flink 檢查點上建立新檔案。在本機執行時，應用程式會依照程式碼中的指定，每 5 秒 (5,000 毫秒) 執行一次檢查點。

```
if (env instanceof LocalStreamEnvironment) {
    env.enableCheckpointing(5000);
}
```

### 瀏覽 S3 儲存貯體並觀察應用程式撰寫的檔案

1. 在開啟 Amazon S3 主控台 <https://console.aws.amazon.com/s3/>。
2. 選擇您先前建立的值區。
3. 導覽至 output 路徑，然後瀏覽至與時區中目前時間對應的日期和小 UTC 時資料夾。
4. 定期刷新以觀察每 5 秒出現一次新文件。
5. 選擇並下載一個文件以觀察內容。



**Note**

依預設，檔案沒有副檔名。內容的格式為JSON。您可以使用任何文本編輯器打開文件以檢查內容。

## 停止應用程式在本機執行

停止在您的IDE。通IDE常會提供「停止」選項。確切的位置和方法取決於IDE。

## 編譯並封裝應用程式程式碼

在本節中，您可以使用阿帕奇 Maven 編譯 Java 代碼並將其打包到一個JAR文件中。您可以使用 Maven 命令行工具或IDE。

若要使用 Maven 命令列進行編譯和封裝

移至包含 Java GettingStarted 專案的目錄，並執行下列命令：

```
$ mvn package
```

要使用您的編譯和打包 IDE

mvn package從你的 IDE Maven 集成運行。

在這兩種情況下，都會target/amazon-msf-java-table-app-1.0.jar建立JAR檔案。

**Note**

從您運行構建項目IDE可能不會創建該JAR文件。

## 上傳應用程式碼JAR檔案

在本節中，您將在上一節中建立的JAR檔案上傳到您在本教學開始時建立的 Amazon S3 儲存貯體。如果您已經完成，請完成[建立 Amazon S3 儲存貯體](#)。

上傳應用程式的程式碼

1. 在開啟 Amazon S3 主控台<https://console.aws.amazon.com/s3/>。

2. 選擇您先前為應用程式程式碼建立的值區。
3. 選擇「上傳」欄位。
4. 選擇 Add files (新增檔案)。
5. 導航到上一節中生成的JAR文件：`target/amazon-msf-java-table-app-1.0.jar`。
6. 選擇「上傳」而不變更任何其他設定。

#### Warning

請確定您在中選取了正確的JAR檔案`<repo-dir>/java/GettingStarted/target/amazon/msf-java-table-app-1.0.jar`。  
目標目錄也包含您不需要上傳的其他JAR檔案。

## 建立並設定 Apache Flink 應用程式的受管理服務

您可以使用主控台或 AWS CLI 在本教程中，您將使用控制台。

#### 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。使用建立應用程式時 AWS CLI，必須個別建立這些資源。

## 建立應用程式

1. 在 <https://console.aws.amazon.com/flink> 開啟適用於阿帕奇 Flink 的受管理服務
2. 確認已選取正確的區域：美國東部 (維吉尼亞北部) us-east-1。
3. 在右側功能表中，選擇 Apache Flink 應用程式，然後選擇 [建立串流應用程式]。或者，在初始頁面的 [開始使用] 區段中選擇 [建立串流應用程式]。
4. 在 [建立串流應用程式] 頁面上，完成下列動作：
  - 對於選擇設定串流處理應用程式的方法，請選擇從頭開始建立。
  - 對於阿帕奇 Flink 配置，應用程序 Flink 版本，請選擇阿帕奇 Flink 1.19。
  - 在 [應用程式設定] 區段中，完成下列步驟：
    - 在應用程式名稱中，輸入 **MyApplication**。
    - 對於 Description (說明)，輸入 **My Java Table API test app**。

- 對於存取應用程式資源，請選擇 [建立/更新IAM角色運動分析 MyApplication--us-east-1] (含必要原則)。
  - 在應用程式設定的範本中，完成下列操作：
    - 對於「範本」，請選擇「開發」。
5. 選擇建立串流應用程式。

#### Note

當您使用主控台為 Apache Flink 應用程式建立受管理服務時，您可以選擇為應用程式建立IAM角色和原則。應用程式使用此角色和政策來存取其相依資源。這些IAM資源會使用您的應用程式名稱和區域命名，如下所示：

- 政策：`kinesis-analytics-service-MyApplication-us-east-1`
- 角色：`kinesisanalytics-MyApplication-us-east-1`

## 編輯IAM策略

編輯IAM策以新增存取 Amazon S3 儲存貯體的許可。

編輯IAM政策以新增 S3 儲存貯體許可

1. 在開啟IAM主控台<https://console.aws.amazon.com/iam/>。
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-east-1** 政策。
3. 選擇 [編輯]，然後選擇索JSON引標籤。
4. 將下列政策範例的反白部分新增至政策。取代範例帳戶 ID (`012345678901`) 使用您的帳戶 ID 和 `<bucket-name>` 使用您建立的 S3 儲存貯體的名稱。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
```

```
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket/kinesis-analytics-placeholder-s3-object"
      ]
    },
    {
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "ListCloudwatchLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
      ]
    },
    {
      "Sid": "PutCloudwatchLogs",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
      ]
    },
    {
      "Sid": "WriteOutputBucket",
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::my-bucket"
      ]
    }
  ]
}
```

```

    ]
  }
}

```

5. 選擇下一步，然後選擇儲存變更。

## 設定應用程式

編輯應用程式以設定應用程式程式碼人工因素。

### 設定應用程式

1. 在 MyApplication 頁面上，選擇設定。
2. 在 [應用程式碼位置] 區段中，選擇 [設定]。
  - 對於 Amazon S3 儲存貯體，請選取先前為應用程式程式碼建立的儲存貯體。選擇 [瀏覽] 並選取正確的值區，然後選擇 [選擇]。請勿按一下值區名稱。
  - 對於 Amazon S3 物件的路徑，請輸入 **amazon-msf-java-table-app-1.0.jar**。
3. 對於 [存取權限]，選擇 [建立/更新IAM角色] kinesis-analytics-MyApplication-us-east-1。
4. 在 [執行階段屬性] 區段中，新增下列屬性。
5. 選擇「新增項目」，然後新增下列每個參數：

| 群組 ID  | 金鑰   | 值                       |
|--------|------|-------------------------|
| bucket | name | <b>your-bucket-name</b> |
| bucket | path | output                  |

6. 請勿修改任何其他設定。
7. 選擇 Save changes (儲存變更)。

### 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

該應用程式現在已配置並準備好運行。

### 執行應用程式

1. 返回適用於 Apache Flink 的 Amazon 受管服務中的主控台頁面，然後選擇MyApplication。
2. 選擇執行以啟動應用程式。
3. 在 [應用程式還原] 組態上，選擇 [以最新快照執行]。
4. 選擇執行。
5. 「應用程式中的狀態」詳細資訊會從Ready應用程式啟動到再轉換為Running之後。Starting

當應用程式處於Running狀態時，您可以開啟 Flink 儀表板。

### 開啟控制面板並檢視工作

1. 選擇「開啟阿帕奇快顯儀表板」。儀表板會在新頁面中開啟。
2. 在「正在執行的工作」清單中，選擇您可以看到的單一工作。

#### Note

如果您設定執行階段屬性或編輯不正確的IAM原則，應用程式狀態可能會變更為Running，但 Flink 儀表板會顯示持續重新啟動的工作。當應用程式設定錯誤或缺少存取外部資源的權限時，這是常見的失敗案例。

發生這種情況時，請檢查 Flink 儀表板中的「例外」標籤，以調查問題的原因。

## 觀察正在運行的應用程式的指標

在MyApplication頁面的 Amazon CloudWatch 指標區段中，您可以看到執行中應用程式中的一些基本指標。

## 若要檢視測量結果

1. 在「重新整理」按鈕旁邊，從下拉式清單中選取 10 秒。
2. 當應用程式執行且健康狀態良好時，您可以看到正常執行時間指標不斷增加。
3. 完整重啟指標應為零。如果正在增加，則配置可能有問題。檢閱 Flink 儀表板上的「例外」標籤，以調查問題。
4. 在運作良好的應用程式中，失敗的檢查點數量應為零。

### Note

此儀表板會顯示一組固定的指標，精細度為 5 分鐘。您可以使用儀表板中的任何指標來建立自訂應用程式 CloudWatch 儀表板。

## 觀察將資料寫入目標值區的應用程式

現在，您可以觀察到在 Amazon 託管服務中運行的 Apache Flink 將文件寫入 Amazon S3 的應用程式。

若要觀察這些檔案，請遵循與檢查應用程式在本機執行時所寫入的檔案相同的程序。請參閱 [觀察將資料寫入 S3 儲存貯體的應用程式](#)。

請記住，應用程式會在 Flink 檢查點上寫入新檔案。在適用於 Apache Flink 的 Amazon 受管服務上執行時，檢查點預設為啟用，並且每 60 秒執行一次。該應用程式大約每 1 分鐘創建一次新文件。

## 停止應用程式

若要停止應用程式，請移至名為的 Apache Flink 應用程式的受管理服務的主控台頁面。MyApplication

### 停止應用程式

1. 從「動作」下拉式清單中，選擇「停止」。
2. 「應用程式中的狀態」詳細資訊Ready會從應Running用程式完全停止轉換為Stopping，然後再轉換為應用程式的

### Note

別忘了也停止將資料從 Python 指令碼或 Kinesis 資料產生器傳送至輸入串流。

## 下一步驟

### [清理 AWS 資源](#)

## 清理 AWS 資源

本節包括清理在入門 (表格API) 教學課程中建立的 AWS 資源的程序。

本主題包含下列章節。

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的IAM資源](#)
- [刪除您的 CloudWatch 資源](#)
- [下一步驟](#)

## 刪除適用於 Apache Flink 應用程式的受管理服務

請使用下列程序刪除應用程式。

### 刪除應用程式

1. 在 <https://console.aws.amazon.com/Kinesis> 處開啟室壁運動主控台。
2. 在適用於 Apache Flink 的受管理服務面板中，選擇MyApplication。
3. 從「動作」下拉式清單中，選擇「刪除」，然後確認刪除。

## 刪除您的 Amazon S3 對象和存儲桶

使用下列程序來刪除 S3 物件和儲存貯體。

### 從 S3 儲存貯體刪除應用程式物件

1. 在開啟 Amazon S3 主控台<https://console.aws.amazon.com/s3/>。
2. 選取您建立的 S3 儲存貯體。
3. 選取您上傳命名的應用程式人工因素amazon-msf-java-table-app-1.0.jar，選擇刪除，然後確認刪除。



## 刪除應用程式寫入的所有輸出檔案

1. 選擇 output 資料夾。
2. 選擇 刪除。
3. 確認您要永久刪除內容。

## 刪除 S3 儲存貯體

1. 在開啟 Amazon S3 主控台<https://console.aws.amazon.com/s3/>。
2. 選取您建立的 S3 儲存貯體。
3. 選擇刪除並確認刪除。

## 刪除您的IAM資源

請使用下列程序刪除IAM源。

若要刪除您的IAM資源

1. 在開啟IAM主控台<https://console.aws.amazon.com/iam/>。
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-ea st-1 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 us-east- 1 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. 在開啟 CloudWatch 主控台<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，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。

## 下一步驟

### [探索其他資源](#)

## 探索其他資源

現在您已經建立並執行使用資料表的 Apache Flink 應用程式的受管理服務API，請參閱[探索其他資源](#)中的[開始使用阿帕奇 Flink 的 Amazon 託管服務 \(\) DataStream API](#)。

# 開始使用適用於 Python 的阿帕奇 Flink 的 Amazon 託管服務

本節將為您介紹使用 Python 和資料表API的 Apache Flink 受管理服務的基本概念。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

- [檢閱 Apache Flink 應用程式的受管理服務元件](#)
- [符合先決條件](#)
- [為 Python 應用程序創建並運行阿帕奇 Flink 的託管服務](#)
- [清理 AWS 資源](#)

## 檢閱 Apache Flink 應用程式的受管理服務元件

### Note

Amazon 阿帕奇 Flink 管理服務支持所有[阿帕奇 F APIs](#) link。視API您選擇的項目而定，應用程式的結構會稍有不同。在 Python 中開發 Apache Flink 應用程序時，一種流行的方法是使用 SQL 嵌入在 Python 代碼中定義應用程序流程。這是我們遵循下面的 Gettgin 開始教程的方法。

為了處理資料，Apache Flink 應用程式的受管理服務會使用 Python 指令碼來定義資料流程，以處理輸入並使用 Apache Flink 執行階段產生輸出。

Apache Flink 應用程序的典型託管服務具有以下組件：

- **執行期屬性**：您可以使用執行期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**：應用程序消耗來自一個或多個來源的數據。來源使用[連接器](#)從外部系統 (例如 Kinesis 資料串流或 Amazon MSK 主題) 讀取資料。您也可以使用特殊連接器從應用程式內產生資料。當您使用時 SQL，應用程式會將來源定義為來源資料表。
- **轉換**：應用程式會使用一或多個可篩選、豐富或彙總資料的轉換來處理資料。當您使用時 SQL，應用程式會將轉換定義為 SQL 查詢。
- **接收器**：應用程序通過接收器將數據發送到外部源。接收器使用[連接器](#)將資料傳送到外部系統，例如 Kinesis 資料串流、Amazon MSK 主題、Amazon S3 儲存貯體或關聯式資料庫。您也可以使用特殊

的連接器來列印輸出，以供開發之用。使用時SQL，應用程式會將接收器定義為插入結果的接收器表格。如需詳細資訊，請參閱[在 Apache Flink 的受管理服務中使用接收器寫入資料](#)。

您的 Python 應用程式也可能需要外部相依性，例如其他 Python 程式庫或應用程式使用的任何 Flink 連接器。當您封裝應用程式時，您必須包含應用程式所需的每個相依性。本教學課程示範如何包含連接器相依性，以及如何封裝應用程式，以便在 Apache Flink 的 Amazon 受管服務上進行部署。

## 符合先決條件

若要完成此自學課程，您必須具備下列項目：

- Python 3.11，[最好使用一個獨立的環境，如 VirtualEnv \( venv \)，康達，或迷你圖標](#)。
- [Git 客戶端](#)-如果您還沒有安裝 Git 客戶端。
- [Java 開發套件 \(JDK\) 版本 11](#)-安裝 Java JDK 11 並將JAVA\_HOME環境變數設定為指向您的安裝位置。如果您沒有 JDK 11，則可以使用[Amazon Corretto](#)或我們選擇JDK的任何標準。
  - 若要確認您已JDK正確安裝，請執行下列命令。如果您使用的是 Amazon Corretto 11 以JDK外的輸出將是不同的。確保該版本是 11.x。

```
$ java --version
```

```
openjdk 11.0.23 2024-04-16 LTS
```

```
OpenJDK Runtime Environment Corretto-11.0.23.9.1 (build 11.0.23+9-LTS)
```

```
OpenJDK 64-Bit Server VM Corretto-11.0.23.9.1 (build 11.0.23+9-LTS, mixed mode)
```

- [阿帕奇 Maven](#) 的-安裝阿帕奇 Maven 的，如果你還沒有這樣做了。如需詳細資訊，請參閱[安裝阿帕奇 Maven](#)。
  - 若要測試您的 Apache Maven 安裝，請使用下列指令：

```
$ mvn -version
```

### Note

雖然你的應用程序是用 Python 編寫的，阿帕奇 Flink 運行在 Java 虛擬機 ( JVM )。它會將大部分的相依性 (例如 Kinesis 連接器) 以檔案形JAR式分配。若要管理這些相依性，並將應用程式封裝在ZIP檔案中，請使用 [Apache Maven](#)。本教程介紹瞭如何做到這一點。

### Warning

我們建議您使用 Python 3.11 進行本機開發。這與 Amazon 管理服務用於阿帕奇 Flink 與 Flink 運行時 1.19 使用的 Python 版本相同。

在 Python 3.12 上安裝元件庫 1.19 可能會失敗。

如果您的計算機上默認安裝了另一個 Python 版本，我們建議您創建一個獨立的環境，例如 VirtualEnv 使用 Python 3.11。

## IDE用於本地發展

我們建議您使用開發環境，例如 [PyCharm](#) 或 [Visual Studio 程式碼](#) 來開發和編譯您的應用程式。

然後，完成以下步驟的前兩個步驟 [開始使用阿帕奇 Flink 的 Amazon 託管服務 \(\) DataStream API](#)：

- [設定 AWS 帳戶並建立管理員使用者](#)
- [設置 AWS Command Line Interface \( AWS CLI \)](#)

若要開始使用，請參閱 [建立應用程式](#)。

## 為 Python 應用程式創建並運行阿帕奇 Flink 的託管服務

在本節中，您將為 Python 應用程式建立 Apache Flink 應用程式的受管理服務，並將 Kinesis 串流作為來源和接收器。

本節包含下列步驟：

- [建立相依資源](#)
- [設定您的本機開發環境](#)
- [下載並檢查阿帕奇 Flink 流 Python 代碼](#)
- [管理JAR相依性](#)
- [將樣本記錄寫入輸入流](#)
- [在本機執行應用程式](#)
- [觀察 Kinesis 串流中的輸入和輸出資料](#)
- [停止應用程式在本機執行](#)
- [Package 您的應用程式碼](#)

- [將應用程式套件上傳到 Amazon S3 儲存貯體](#)
- [建立並設定 Apache Flink 應用程式的受管理服務](#)
- [下一步驟](#)

## 建立相依資源

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

- 兩個 Kinesis 串流，用於輸入和輸出。
- 用於存放應用程式程式碼的 Amazon S3 儲存貯體。

### Note

本教學課程假設您正在 us-east-1 區域中部署應用程式。如果您使用其他區域，則必須相應地調整所有步驟。

## 建立兩個 Kinesis 串流

在為本練習建立適用於 Apache Flink 應用程式的受管服務之前，請在您將用來部署應用程式的相同區域中建立兩個 Kinesis 資料串流 (ExampleInputStream 和 ExampleOutputStream) (在此範例中為 us-east-1)。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)，請使用下列 Amazon Kinesis create-stream AWS CLI 指令。

```
$ aws kinesys create-stream \  
--stream-name ExampleInputStream \  
--shard-count 1 \  
--region us-east-1
```

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-east-1
```

## 建立 Amazon S3 儲存貯體

您可以使用主控台建立 Amazon S3 儲存貯體。如需建立這些資源的相關指示，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](#)。為 Amazon S3 儲存貯體指定一個全域唯一的名稱，例如透過附加您的登入名稱。

### Note

請務必在您用於本教學課程 (us-east-1) 的區域中建立 S3 儲存貯體。

## 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/<my-application> 的日誌群組。
- 名為 kinesis-analytics-log-stream 的日誌串流。

## 設定您的本機開發環境

對於開發和調試，您可以在計算機上運行 Python Flink 應用程序。您可以使用 `python main.py` 或在您選擇 IDE 的 Python 中從命令行啟動應用程序。

### Note

在你的開發機器上，你必須有 Python 3.10 或 3.11，Java 11，阿帕奇 Maven 和 Git 安裝。我們建議您使用 IDE 如 [PyCharm](#) 或 [視覺工作室程式碼](#)。若要確認您符合所有先決條件，請參閱[繼續滿足完成練習的先決條件](#)之前。

## 安裝 PyFlink 式庫

若要開發應用程式並在本機執行，您必須安裝 Flink Python 程式庫。

1. 使用 VirtualEnv，康達或任何類似的 Python 工具創建一個獨立的 Python 環境。
2. 在該環境中安裝 PyFlink 程式庫。使用您將在阿帕奇 Flink 的 Amazon 受管服務中使用的相同 Apache Flink 執行階段版本。目前，建議的執行階段為 1.19.1。

```
$ pip install apache-flink==1.19.1
```

3. 當您執行應用程式時，請確定環境處於作用中狀態。如果您在中執行應用程式IDE，請確定IDE使用環境做為執行階段。該過程取決於您IDE正在使用的。

### Note

您只需要安裝 PyFlink 式庫即可。您不需要在您的機器上安裝 Apache Flink 叢集。

## 驗證您的 AWS 會話

應用程式會使用 Kinesis 資料串流來發佈資料。在本機執行時，您必須擁有具有寫入 Kinesis 資料串流之權限的有效 AWS 驗證工作階段。請使用下列步驟來驗證您的工作階段：

1. 如果您沒有設定 AWS CLI 具有有效認證的具名設定檔，請參閱[設置 AWS Command Line Interface \(AWS CLI\)](#)。
2. 發佈下列測試記錄，確認您的設定正確，且您的使用者具有寫入 Kinesis 資料串流的權限：AWS CLI

```
$ aws kinesis put-record --stream-name ExampleOutputStream --data TEST --partition-key TEST
```

3. 如果您IDE有要整合的外掛程式 AWS，您可以使用該外掛程式將憑證傳遞至在IDE。如需詳細資訊，請參閱[AWS 適用於 PyCharm Visual Studio 程式碼的AWS 工具AWS組和 IntelliJ IDEA 的工具組](#)。

## 下載並檢查阿帕奇 Flink 流 Python 代碼

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. 使用以下指令複製遠端儲存庫：

```
git clone https://github.com/aws-samples/amazon-managed-service-for-apache-flink-examples.git
```

2. 導覽至 `./python/GettingStarted` 目錄。

## 檢閱應用程式元

應用程式程式碼位於 `main.py`。我們使用 SQL 嵌入在 Python 中來定義應用程序的流程。

### Note

為了獲得最佳化的開發人員體驗，應用程式的設計目的是在適用於 Apache Flink 的 Amazon 受管服務和本機上進行開發時，不需要變更任何程式碼即可執行。應用程式會使用環境變數 `IS_LOCAL = true` 來偵測何時在本機執行。您必須在 shell 上 `IS_LOCAL = true` 或 IDE 的運行配置中設置環境變量

- 應用程式會設定執行環境並讀取執行階段組態。若要同時在適用於 Apache Flink 和本機的 Amazon 受管服務上運作，應用程式會檢查 `IS_LOCAL` 變數。
- 以下是應用程式在適用於 Apache Flink 的 Amazon 受管服務中執行時的預設行為：
  1. 載入與應用程式封裝的相依性。有關更多信息，請參閱 ( [鏈接](#) )
  2. 從您在 Apache Flink 應用程式的 Amazon 受管服務中定義的執行階段屬性載入組態。有關更多信息，請參閱 ( [鏈接](#) )
- 當應用程式偵測到您在本機執行應用程式 `IS_LOCAL = true` 時：
  1. 從專案載入外部相依性。
  2. 從專案中包含的 `application_properties.json` 檔案載入規劃。

```
...
APPLICATION_PROPERTIES_FILE_PATH = "/etc/flink/application_properties.json"
...
is_local = (
    True if os.environ.get("IS_LOCAL") else False
)
...
if is_local:
    APPLICATION_PROPERTIES_FILE_PATH = "application_properties.json"
```

```

CURRENT_DIR = os.path.dirname(os.path.realpath(__file__))
table_env.get_config().get_configuration().set_string(
    "pipeline.jars",
    "file:/// " + CURRENT_DIR + "/target/pyflink-dependencies.jar",
)

```

- 應用程式會使用 [Kinesis 連接器](#) 來定義含 CREATE TABLE 陳述式的來源資料表。此表格會從輸入 Kinesis 串流讀取資料。該應用程式從運行時配置獲取流的名稱，區域和初始位置。

```

table_env.execute_sql(f"""
    CREATE TABLE prices (
        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' = '{input_stream_name}',
        'aws.region' = '{input_stream_region}',
        'format' = 'json',
        'json.timestamp-format.standard' = 'ISO-8601'
    ) """)

```

- 在此範例中，應用程式也會使用 [Kinesis 連接器來定義接收表](#)。這個故事將數據發送到輸出 Kinesis 流。

```

table_env.execute_sql(f"""
    CREATE TABLE output (
        ticker VARCHAR(6),
        price DOUBLE,
        event_time TIMESTAMP(3)
    )
    PARTITIONED BY (ticker)
    WITH (
        'connector' = 'kinesis',
        'stream' = '{output_stream_name}',
        'aws.region' = '{output_stream_region}',
        'sink.partitioner-field-delimiter' = ';',
        'sink.batch.max-size' = '100',
        'format' = 'json',
        'json.timestamp-format.standard' = 'ISO-8601'
    )

```

```
)""")
```

- 最後，應用程式會從來源資料表中執行接收器表格。SQL INSERT INTO...在更複雜的應用程式中，在寫入接收器之前，您可能還有其他步驟來轉換資料。

```
table_result = table_env.execute_sql("""INSERT INTO output
    SELECT ticker, price, event_time FROM prices""")
```

- 您必須在函數結尾新增另一個步驟，才main()能在本機執行應用程式：

```
if is_local:
    table_result.wait()
```

如果沒有這個陳述式，應用程式會在您在本機執行時立即終止。當您在 Apache Flink 的 Amazon 受管服務中執行應用程式時，不得執行此陳述式。

## 管理JAR相依性

一個 PyFlink 應用程序通常需要一個或多個連接器。本教學課程中的應用程式使用 [Kinesis 連接器](#)。由於 Apache Flink 在 Java 中執行 JVM，因此無論您是否在 Python 中實作應用程式，連接器都會以 JAR 檔案的形式散佈。在適用於 Apache Flink 的 Amazon 受管服務上部署應用程式時，必須將這些相依性與應用程式封裝。

在這個例子中，我們展示了如何使用 Apache 的 Maven 獲取依賴關係和打包應用程序在 Apache Flink 託管服務上運行。

### Note

還有其他方法來獲取和打包依賴關係。此範例示範可正確搭配一個或多個連接器運作的方法。它也可讓您在本地、開發和 Apache Flink 受管理服務上執行應用程式，而無需變更程式碼。

## 使用 pom.xml 檔案

阿帕奇 Maven 使用該 pom.xml 文件來控制依賴關係和應用程序打包。

任何 JAR 從屬關係都在 <dependencies>...</dependencies> 塊中的 pom.xml 文件中指定。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  ...
  <dependencies>
    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-connector-kinesis</artifactId>
      <version>4.3.0-1.19</version>
    </dependency>
  </dependencies>
  ...
```

若要尋找要使用的正確加工品和連接器版本，請參閱[搭配 Apache Flink 的受管理服務使用 Apache Flink 連接器](#)。請確定您參考的是您正在使用的 Apache Flink 版本。在此範例中，我們使用 Kinesis 連接器。對於阿帕奇 Flink 1.19，連接器版本為。4.3.0-1.19

#### Note

如果您使用的是 Apache Flink 1.19，則沒有專門針對此版本發行的連接器版本。使用針對 1.18 版發行的連接器。

## 下載和封裝相依性

使用 Maven 下載 pom.xml 檔案中定義的相依性，並將它們封裝給 Python Flink 應用程式。

1. 瀏覽至包含名為 Python 入門專案的目錄 python/GettingStarted。
2. 執行以下命令：

```
$ mvn package
```

Maven 創建一個名為的新文件 ./target/pyflink-dependencies.jar。當您在電腦上進行本機開發時，Python 應用程式會尋找此檔案。

#### Note

如果您忘記運行此命令，當您嘗試運行應用程序時，它將失敗並顯示錯誤：找不到標識符「kinesis 的任何工廠。

## 將樣本記錄寫入輸入流

在本節中，您會將範例記錄傳送至串流以供應用程式處理。您有兩個選項可以使用 Python 指令碼或 [Kinesis 資料產生器](#) 來產生範例資料。

### 使用 Python 腳本生成示例數據

您可以使用 Python 腳本將示例記錄發送到流。

#### Note

要運行這個 Python 腳本，你必須使用 Python 3.x 和有 [Python \( 博托 \)](#) 庫安裝。AWS SDK

若要開始將測試資料傳送至 Kinesis 輸入串流：

1. 從數據生成器 [GitHub 存儲庫](#) 下載數據生成器 `stock.py` Python 腳本。
2. 執行 `stock.py` 指令碼：

```
$ python stock.py
```

當您完成教學課程的其餘部分時，請保持指令碼的執行狀態。您現在可以執行您的 Apache Flink 應用程式。

### 使用 Kinesis 資料產生器產生範例資料

或者，若要使用 Python 指令碼，您也可以使用 [託管版本](#) 提供的 [Kinesis 資料產生器](#)，將隨機範例資料傳送至串流。Kinesis 資料產生器會在您的瀏覽器中執行，而且您不需要在電腦上安裝任何項目。

若要設定和執行 Kinesis 資料產生器：

1. 請遵循 [Kinesis 資料產生器文件](#) 中的指示來設定工具的存取權。您將執行設定使用者和密碼的 AWS CloudFormation 範本。
2. 透過 CloudFormation 範本產生的存取 Kinesis 資料 URL 產生器。CloudFormation 模板完成後，您可以 URL 在「輸出」選項卡中找到。
3. 設定資料產生器：
  - 區域：選取您在此教學課程中使用的區域：us-east-1
  - 串流/交付串流：選取應用程式將使用的輸入串流：ExampleInputStream

- 每秒記錄數:100
- 記錄範本：複製並貼上下列範本：

```
{
  "event_time" : "{{date.now("YYYY-MM-DDTkk:mm:ss.SSSSS")}}",
  "ticker" : "{{random.arrayElement(
    ["AAPL", "AMZN", "MSFT", "INTC", "TBV"]
  )}}",
  "price" : {{random.number(100)}}
}
```

4. 測試範本：選擇「測試範本」，然後確認產生的記錄與下列內容類似：

```
{ "event_time" : "2024-06-12T15:08:32.04800", "ticker" : "INTC", "price" : 7 }
```

5. 啟動資料產生器：選擇「選取傳送資料」。

Kinesis 資料產生器現在正在將資料傳送至 `ExampleInputStream`

## 在本機執行應用程式

您可以在本地測試應用程序，從命令行運行 `python main.py` 或從 IDE。

若要在本機執行應用程式，您必須安裝正確版本的程式 PyFlink 庫，如上一節所述。有關更多信息，請參閱 ( [鏈接](#) )

### Note

在繼續之前，請確認輸入和輸出串流是否可用。請參閱 [建立兩個 Amazon Kinesis 資料串流](#)。另外，請確認您具有從兩個串流讀取和寫入的權限。請參閱 [驗證您的 AWS 會話](#)。

## 將 Python 項目導入到您的 IDE

若要開始在您的應用程式上工作 IDE，您必須將它匯入為 Python 專案。

您複製的儲存庫包含多個範例。每個例子都是一個單獨的項目。在本教學課程中，請將 `./python/GettingStarted` 子目錄中的內容匯入您的 IDE。

將程式碼匯入為現有的 Python 專案。

**Note**

導入新 Python 項目的確切過程取決於IDE您使用的項目。

## 檢查本地應用程序配置

在本機執行時，應用程式會使用下專application\_properties.json案資源資料夾中檔案中的組態./src/main/resources。您可以編輯此檔案以使用不同的 Kinesis 串流名稱或區域。

```
[
  {
    "PropertyGroupId": "InputStream0",
    "PropertyMap": {
      "stream.name": "ExampleInputStream",
      "flink.stream.initpos": "LATEST",
      "aws.region": "us-east-1"
    }
  },
  {
    "PropertyGroupId": "OutputStream0",
    "PropertyMap": {
      "stream.name": "ExampleOutputStream",
      "aws.region": "us-east-1"
    }
  }
]
```

## 在本機執行 Python 應用程式

您可以在本機執行應用程式，無論是從命令列做為一般 Python 指令碼，或從IDE。

### 若要從命令列執行應用程式

1. 請確定獨立 Python 環境 (例如 Conda) 或您安裝 Python Flink 程式庫的 VirtualEnv 位置目前處於作用中狀態。
2. 確保您至mvn package少跑了一次。
3. 設定 IS\_LOCAL = true 環境變數：

```
$ export IS_LOCAL=true
```

#### 4. 運行應用程式作為一個常規的 Python 腳本。

```
$python main.py
```

若要從中執行應用程式 IDE

1. 配置您IDE使用以下配置運行main.py腳本：
  1. 使用獨立的 Python 環境，例如 Conda 或您安裝程式 PyFlink 庫的 VirtualEnv 位置。
  2. 使用 AWS 認證來存取輸入和輸出 Kinesis 資料串流。
  3. 設定 IS\_LOCAL = true。
2. 設置運行配置的確切過程取決於您的IDE和變化。
3. 設置完畢後IDE，請運行 Python 腳本並使用應用程式運行IDE時提供的工具。

#### 在本機檢查應用程式

在本地運行時，除了應用程式啟動時打印和顯示的幾行之外，應用程式不會在控制台中顯示任何日誌。PyFlink 將記錄寫入安裝 Python Flink 程式庫的目錄中的檔案中。應用程式會在啟動時列印記錄檔的位置。您也可以執行下列命令來尋找記錄檔：

```
$ python -c "import pyflink;import os;print(os.path.dirname(os.path.abspath(pyflink.__file__))+'/log')"
```

1. 列出記錄目錄中的檔案。您通常會找到一個.log文件。
2. 在應用程式運行時尾部文件：`tail -f <log-path>/<log-file>.log`。

#### 觀察 Kinesis 串流中的輸入和輸出資料

您可以使用 Amazon Kinesis Kinesis 主控台內的資料檢視器，觀察 (產生範例 Python) 或 Kinesis 資料產生器 (連結) 傳送至輸入串流的記錄。

要觀察記錄：

#### 停止應用程式在本機執行

停止在您的IDE. 通IDE常會提供「停止」選項。確切的位置和方法取決於IDE.



## Package 您的應用程式碼

在本節中，您可以使用 Apache Maven 將應用程式程式碼和所有必要的相依性封裝在 .zip 檔案中。

再次運行 Maven 包命令：

```
$ mvn package
```

這個命令會產生檔案 `target/managed-flink-pyflink-getting-started-1.0.0.zip`。

## 將應用程式套件上傳到 Amazon S3 儲存貯體

在本節中，您將在上一節中建立的 .zip 檔案上傳到您在本教學開始時建立的 Amazon 簡單儲存服務 (Amazon S3) 儲存貯體。如果您尚未完成此步驟，請參閱 ( [鏈接](#) )。

上傳應用程式程式碼JAR檔案

1. 在開啟 Amazon S3 主控台 <https://console.aws.amazon.com/s3/>。
2. 選擇您先前為應用程式程式碼建立的值區。
3. 選擇上傳。
4. 選擇 Add files (新增檔案)。
5. 導覽至在上一個步驟中產生的 .zip 檔案：`target/managed-flink-pyflink-getting-started-1.0.0.zip`。
6. 選擇「上傳」而不變更任何其他設定。

## 建立並設定 Apache Flink 應用程式的受管理服務

您可以使用主控台或 AWS CLI 在本教程中，我們將使用控制台。

建立應用程式

1. 在 <https://console.aws.amazon.com/flink> 開啟適用於阿帕奇 Flink 的受管理服務
2. 確認已選取正確的區域：美國東部 (維吉尼亞北部) us-east-1。
3. 開啟右側功能表並選擇 Apache Flink 應用程式，然後選擇建立串流應用程式。或者，從初始頁面的 [開始使用] 區段中選擇 [建立串流應用程式]。
4. 在 [建立串流應用程式] 頁面上：

- 對於選擇設定串流處理應用程式的方法，請選擇從頭開始建立。
- 對於阿帕奇 Flink 配置，應用程式 Flink 版本，請選擇阿帕奇 Flink 1.19。
- 對於應用程式配置：
  - 在應用程式名稱中，輸入 **MyApplication**。
  - 對於 Description (說明)，輸入 **My Python test app**。
  - 在 [存取應用程式資源] 中，選擇 [建立/更新IAM角色運動分析 MyApplication--us-east-1 具有必要原則]。
- 對於應用程式設定的範本：
  - 針對範本，選擇開發。
- 選擇建立串流應用程式。

#### Note

當您使用主控台為 Apache Flink 應用程式建立受管理服務時，您可以選擇為應用程式建立IAM角色和原則。應用程式使用此角色和政策來存取其相依資源。這些IAM資源會使用您的應用程式名稱和區域命名，如下所示：

- 政策：`kinesis-analytics-service-MyApplication-us-west-2`
- 角色：`kinesisanalytics-MyApplication-us-west-2`

Amazon 阿帕奇 Flink 託管服務以前被稱為 Kinesis Data Analytics。kinesis-analytics 為了回溯相容，自動產生的資源名稱會加上前置詞。

## 編輯IAM策略

編輯IAM策略以新增存取 Amazon S3 儲存貯體的許可。

編輯IAM政策以新增 S3 儲存貯體許可

1. 在開啟IAM主控台<https://console.aws.amazon.com/iam/>。
2. 選擇政策。選擇主控台為您在上節所建立的 **kinesis-analytics-service-MyApplication-us-east-1** 政策。
3. 選擇 [編輯]，然後選擇索引標籤。

4. 將下列政策範例的反白部分新增至政策。取代範例帳戶 IDs (*012345678901*) 使用您的帳戶 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3::my-bucket/kinesis-analytics-placeholder-s3-object"
      ]
    },
    {
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "ListCloudwatchLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
      ]
    },
    {
      "Sid": "PutCloudwatchLogs",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],

```

```

    "Resource": [
      "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleOutputStream"
  }
]
}

```

5. 選擇下一步，然後選擇儲存變更。

## 設定應用程式

編輯應用程式組態以設定應用程式程式碼人工因素。

### 設定應用程式

1. 在MyApplication頁面上，選擇設定。
2. 在應用程式程式碼位置區段中：
  - 對於 Amazon S3 儲存貯體，請選取先前為應用程式程式碼建立的儲存貯體。選擇「瀏覽」並選取正確的值區，然後選擇「選擇」。請勿選取值區名稱。
  - 對於 Amazon S3 物件的路徑，請輸入 **managed-flink-pyflink-getting-started-1.0.0.zip**。
3. 對於 [存取權限]，選擇 [建立/更新kinesis-analytics-MyApplication-us-east-1具有必要策略的IAM角色]
4. 移至「執行階段」屬性，並保留所有其他設定的預設值。

5. 選擇「新增項目」，然後新增下列每個參數：

| 群組 ID  | 金鑰                                | 值   |
|--|-----------------------------------|---|
| <code>InputStream0</code>                        | <code>stream.name</code>          | <code>ExampleInputStream</code>           |
| <code>InputStream0</code>                        | <code>flink.stream.initpos</code> | <code>LATEST</code>                       |
| <code>InputStream0</code>                        | <code>aws.region</code>           | <code>us-east-1</code>                    |
| <code>OutputStream0</code>                       | <code>stream.name</code>          | <code>ExampleOutputStream</code>          |
| <code>OutputStream0</code>                       | <code>aws.region</code>           | <code>us-east-1</code>                    |
| <code>kinesis.analytics.flink.run.options</code> | <code>python</code>               | <code>main.py</code>                      |
| <code>kinesis.analytics.flink.run.options</code> | <code>jarfile</code>              | <code>lib/pyflink-dependencies.jar</code> |

6. 請勿修改任何其他區段，然後選擇 [儲存變更]。

#### 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

該應用程式現在已配置並準備好運行。

### 執行應用程式

1. 在適用於 Apache Flink 的 Amazon 受管服務的主控台上，選擇我的應用程式，然後選擇執行。

2. 在下一頁的 [應用程式還原組態] 頁面上，選擇 [以最新快照執行]，然後選擇 [執行]。

「應用程式中的狀態」詳細資訊會從Ready應用程式啟動到再轉換為應用程式的Running時間。Starting

當應用程式處於狀Running態時，您現在可以開啟 Flink 儀表板。

### 開啟 儀表板

1. 選擇「開啟阿帕奇 Flink 儀表板」。儀表板會在新頁面上開啟。
2. 在「執行工作」清單中，選擇您可以看到的單一工作。

#### Note

如果您設定執行階段屬性或編輯不正確的IAM原則，應用程式狀態可能會變成Running，但 Flink 儀表板會顯示工作正在持續重新啟動。如果應用程式設定錯誤或缺少存取外部資源的權限，這是常見的失敗案例。

發生這種情況時，請檢查 Flink 儀表板中的「例外」標籤以查看問題的原因。

### 觀察正在運行的應用程序的指標

在MyApplication頁面的 Amazon CloudWatch 指標區段中，您可以看到執行中應用程式中的一些基本指標。

#### 若要檢視測量結果

1. 在「重新整理」按鈕旁邊，從下拉式清單中選取 10 秒。
2. 當應用程式執行且健康狀態良好時，您可以看到正常執行時間指標不斷增加。
3. 完整重新啟動指標應為零。如果正在增加，則配置可能有問題。若要調查問題，請檢閱 Flink 儀表板上的「例外」標籤。
4. 在運作狀況良好的應用程式中，失敗的檢查點數量應為零。

#### Note

此儀表板會顯示一組固定的指標，精細度為 5 分鐘。您可以使用儀表板中的任何指標來建立自訂應用程式 CloudWatch 儀表板。

## 觀察 Kinesis 串流中的輸出資料

確定您仍在使用 Python 指令碼或 Kinesis 資料產生器將資料發佈至輸入。

您現在可以使用中的「資料檢視器」，觀察在 Apache Flink 受管理服務上執行之應用程式的輸出 <https://console.aws.amazon.com/kinesis/>，與您之前已執行的操作類似。

若要檢視輸出

1. 在 <https://console.aws.amazon.com/Kinesis> 處開啟室壁運動主控台。
2. 確認區域與您用來執行此自學課程的區域相同。依預設，它是美國東部-1 美國東部 (維吉尼亞北部)。如有必要，請變更「區域」。
3. 選擇「資料串流」。
4. 選取您要觀察的串流。在本教學課程中，使用 ExampleOutputStream。
5. 選擇「資料檢視器」標籤。
6. 選取任何 [碎片]，將 [最新] 保留為起始位置，然後選擇 [取得記錄]。您可能會看到「找不到此要求的記錄」錯誤訊息。如果是這樣，請選擇重試取得記錄。發佈至串流的最新記錄會顯示。
7. 在「資料」欄中選取值，以JSON格式檢查記錄的內容。

## 停止應用程式

若要停止應用程式，請移至名為的 Apache Flink 應用程式的受管理服務的主控台頁面。MyApplication

停止應用程式

1. 從「動作」下拉式清單中，選擇「停止」。
2. 「應用程式中的狀態」詳細資訊Ready會從應Running用程式完全停止轉換為Stopping，然後再轉換為應用程式的

### Note

別忘了也停止將資料從 Python 指令碼或 Kinesis 資料產生器傳送至輸入串流。

## 下一步驟

[清理 AWS 資源](#)

## 清理 AWS 資源

本節包括清理在入門 (Python) 教學課程中建立的 AWS 資源的程序。

本主題包含下列章節。

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

### 刪除適用於 Apache Flink 應用程式的受管理服務

請使用下列程序刪除應用程式。

刪除應用程式

1. 在 <https://console.aws.amazon.com/Kinesis> 處開啟室壁運動主控台。
2. 在適用於 Apache Flink 的受管理服務面板中，選擇 MyApplication。
3. 從「動作」下拉式清單中，選擇「刪除」，然後確認刪除。

### 刪除您的 Kinesis 資料串流

1. 在 <https://console.aws.amazon.com/flink> 開啟適用於阿帕奇 Flink 的受管理服務
2. 選擇 [資料串流]。
3. 選取您建立的兩個串流，ExampleInputStream 和 ExampleOutputStream。
4. 從「動作」下拉式清單中，選擇「刪除」，然後確認刪除。

### 刪除您的 Amazon S3 對象和存儲桶

使用下列程序來刪除 S3 物件和儲存貯體。

若要從 S3 儲存貯體刪除物件

1. 在開啟 Amazon S3 主控台 <https://console.aws.amazon.com/s3/>。



2. 選取您為應用程式成品建立的 S3 儲存貯體。
3. 選取您上傳的應用程式人工因素，命名為amazon-msf-java-stream-app-1.0.jar。
4. 選擇刪除並確認刪除。

### 刪除 S3 儲存貯體

1. 在開啟 Amazon S3 主控台<https://console.aws.amazon.com/s3/>。
2. 選取您為成品建立的值區。
3. 選擇刪除並確認刪除。

#### Note

S3 儲存貯體必須是空的才能刪除它。

## 刪除您的IAM資源

請使用下列程序刪除資IAM源。

若要刪除您的IAM資源

1. 在開啟IAM主控台<https://console.aws.amazon.com/iam/>。
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-east-1 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 us-east-1 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. 在開啟 CloudWatch 主控台 <https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，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。

# 開始使用 (斯卡拉)

## Note

從版本 1.15 開始，Flink 是免費的斯卡拉。應用程式現在可以使用任何 Scala 版本的 Java。Flink 仍然在內部的幾個關鍵組件中使用 Scala，但不會將 Scala 暴露給用戶代碼類加載器。因此，您必須將 Scala 依賴關係添加到您的 JAR-archives 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，您會以 Kinesis 串流做為來源和接收器的 Scala 建立 Apache Flink 應用程式的受管理服務。

本主題包含下列章節：

- [建立相依資源](#)
- [將樣本記錄寫入輸入流](#)
- [下載並檢查應用程式程式碼](#)
- [編譯和上傳應用程式的程式碼](#)
- [創建並運行應用程序 \(控制台\)](#)
- [創建並運行應用程序 \(CLI\)](#)
- [清理 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 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。如需建立這些資源的相關指示，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)，請使用下列 Amazon Kinesis 建立 AWS CLI 串流命令。

```
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

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。

## 將樣本記錄寫入輸入流

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 AWS CLI 將您的配置為使用您的帳戶憑據和默認區域。若要設定您的 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. 導覽至 `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。會讀取執行期應用程式的屬性，來設定連接器。如需執行期屬性的詳細資訊，請參閱[執行期屬性](#)。

## 編譯和上傳應用程式的程式碼

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](#)構建工具來構建應用程序的 Scala 代碼。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 開發工具包 (JDK)。請參閱[完成練習的先決條件](#)。

1. 若要使用應用程式程式碼，請將其編譯並封裝到 JAR 檔案中。您可以使用以下命令編譯和打包代碼 SBT：

```
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 Next (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。

## 創建並運行應用程式 ( 控制台 )

依照以下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 <https://console.aws.amazon.com/flink> 開啟適用於阿帕奇 Flink 的受管理服務
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**。
  - 對於執行期，選擇 Apache Flink。
  - 保持阿帕奇 FLINK 版本 1.19.1 版本的版本。
4. 對於 [存取權限]，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

當您使用主控台為 Apache Flink 應用程式建立受管理服務時，您可以選擇為應用程式建立IAM角色和原則。應用程式使用此角色和政策來存取其相依資源。這些IAM資源會使用您的應用程式名稱和區域命名，如下所示：



- 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 `getting-started-scala-1.0.jar.`。
3. 在 [存取應用程式資源] 下，針對 [存取權限] 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                               |
|---------------------------------------|-----------------------------------|---------------------------------|
| <code>ConsumerConfigProperties</code> | <code>input.stream.name</code>    | <code>ExampleInputStream</code> |
| <code>ConsumerConfigProperties</code> | <code>aws.region</code>           | <code>us-west-2</code>          |
| <code>ConsumerConfigProperties</code> | <code>flink.stream.initpos</code> | <code>LATEST</code>             |

選擇 Save (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                          |
|---------------------------------|---------------------------|----------------------------|
| <b>ProducerConfigProperties</b> | <b>output.stream.name</b> | <b>ExampleOutputStream</b> |
| <b>ProducerConfigProperties</b> | <b>aws.region</b>         | <b>us-west-2</b>           |

- 在監控下，確保監控指標層級設為應用程式。
- 對於CloudWatch 記錄，請選擇啟用核取方塊。
- 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策略

編輯IAM策以新增存取 Amazon S3 儲存貯體的許可。

編輯IAM政策以新增 S3 儲存貯體許可

- 在開啟IAM主控台<https://console.aws.amazon.com/iam/>。
- 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。
- 在摘要頁面，選擇編輯政策。選擇索JSON引標籤。
- 將下列政策範例的反白部分新增至政策。取代範例帳戶 IDs (*012345678901*) 使用您的帳戶 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"
    }
]
}
```

## 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。

## 創建並運行應用程式 ( CLI )

在本節中，您可 AWS Command Line Interface 以使用建立和執行 Apache Flink 應用程式的受管理服務。使用指 AWS CLI 令為 Apache Flink 應用程式建立受管理的服務，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，您的應用程式將無法存取其資料和記錄串流。

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。接著，您可以將原則附加至 IAM 角色 (您在下一節中建立)。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**。將 Amazon 資源名稱 (ARNs) 中的帳戶 ID 取代為您 (**012345678901**) 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"
}
]
```

如需建立權限原則的指 step-by-step 示，請參閱《IAM使用指南》中的[教學課程：建立並附加您的第一個客戶管理原則](#)。

## 建立IAM策略

在本節中，您將建立 Apache Flink 應用程式的受管理服務可假定讀取來源串流並寫入接收串流的IAM角色。

Managed Service for Apache Flink 沒有許可，無法存取串流。您可以透過IAM角色授與這些權限。每個IAM角色都附加了兩個策略。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。

## 建立 IAM 角色

1. 在開啟IAM主控台<https://console.aws.amazon.com/iam/>。
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. 選擇下一步：許可。
7. 在連接許可政策頁面，選擇下一步：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，您已經創建了一個名為的新IAM角色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，選擇許可標籤。
- b. 選擇連接政策。
- 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。記下新角色ARN的。

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

將以下JSON代碼保存到名為的文件中`create_request.json`。將範例角ARN色取代ARN為您先前建立的角色。將值區ARN尾碼 (使用者名稱) 取代為您上一節中選擇的尾碼。使用您的帳戶 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_19",
  "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"  
  }  
]  
}
```

[CreateApplication](#) 使用以下請求執行以創建應用程式：

```
aws kinesisanalyticstv2 create-application --cli-input-json file://create_request.json
```

應用程式現在已建立。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，您將使用 [StartApplication](#) 動作來啟動應用程式。

啟動應用程式

1. 將以下JSON代碼保存到名為的文件中 `start_request.json`。

```
{  
  "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 kinesisanalyticstv2 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

在本節中，您可以使用 [StopApplication](#) 動作來停止應用程式。

## 停止應用程式

1. 將以下JSON代碼保存到名為的文件中`stop_request.json`。

```
{
  "ApplicationName": "s3_sink"
}
```

2. 使用前述請求執行 `StopApplication` 動作以停止應用程式：

```
aws kinesisanalyticstv2 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，而無需重新編譯應用程式程式碼。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. 將以下JSON代碼保存到名為的文件中`update_properties_request.json`。

```
{
  "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 kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

## 更新應用程式的程式碼

當您需要使用新版本的程式碼套件更新應用程式程式碼時，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。如需如何使用 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": "SAMPLEUehYngP87ex1nzYIGYgfhyvDU"
        }
      }
    }
  }
}
```

## 清理 AWS 資源

本節包括清除在「暫停視窗」教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)
- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的 IAM 資源](#)
- [刪除您的 CloudWatch 資源](#)

### 刪除適用於 Apache Flink 應用程式的受管理服務

1. 在 <https://console.aws.amazon.com/flink> 開啟適用於阿帕奇 Flink 的受管理服務
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，選擇刪除，然後確認刪除。

### 刪除您的 Kinesis 資料串流

1. 在 <https://console.aws.amazon.com/Kinesis> 處開啟室壁運動主控台。
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。

## 刪除您的 Amazon S3 對象和存儲桶

1. 在開啟 Amazon S3 主控台<https://console.aws.amazon.com/s3/>。
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。

## 刪除您的IAM資源

1. 在開啟IAM主控台<https://console.aws.amazon.com/iam/>。
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-we st-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 us-west-2 角色。
8. 選擇刪除角色，然後確認刪除。

## 刪除您的 CloudWatch 資源

1. 在開啟 CloudWatch 主控台<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，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。

# 為 Apache Flink 應用程式使用具有託管服務的 Apache 光束

## Note

阿帕奇梁不支持在阿帕奇 Flink 1.19 版本。截至 2024 年 6 月 27 日，沒有適用於 FLINK 1.18 的相容阿帕奇 FLINK 亞軍。如需詳細資訊，請參閱 Apache Beam 文件中的 [Flink 版本相容性](#)。 >

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。

如需如何在 Managed Service for Apache Flink 中使用 Apache Beam 的教學課程，請參閱 [搭 CloudFormation 配 Apache Flink 的受管理服務搭配使用](#)。

本主題包含下列章節：

- [阿帕奇 Flink 運行器與 Apache Flink 管理服務的局限性](#)
- [阿帕奇梁功能與阿帕奇 Flink 管理服務](#)
- [創建使用阿帕奇梁的應用程序](#)

## 阿帕奇 Flink 運行器與 Apache Flink 管理服務的局限性

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 for 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。

## 阿帕奇梁功能與阿帕奇 Flink 管理服務

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。

## 創建使用阿帕奇梁的應用程式

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 搭配使用的相關資訊，請參閱 [為 Apache Flink 應用程式使用具有託管服務的 Apache 光束](#)。

### Note

若要設定此練習的必要先決條件，請先完成 [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#) 練習。

本主題包含下列章節：

- [建立相依資源](#)
- [將樣本記錄寫入輸入流](#)
- [下載並檢查應用程式程式碼](#)
- [編譯應用程式程式碼](#)
- [上傳阿帕奇 Flink 流 Java 代碼](#)
- [建立並執行 Apache Flink 應用程式的受管理服務](#)
- [清理 AWS 資源](#)
- [後續步驟](#)

## 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。如需建立這些資源的相關指示，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。

## 將樣本記錄寫入輸入流

在本節中，您會透過 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. 導覽至 `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 通過調用 `ParDo` 用名為的自定義轉換函數 `PingPongFn` 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，應用程式會在執行期損毀。

```
<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` 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，請先安裝。如需詳細資訊，請參閱[教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#)教學課程中的[完成必要的先決條件](#)。
2. 使用下列命令編譯應用程式：

```
mvn package -Dflink.version=1.15.2 -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)。

## 上傳阿帕奇 Flink 流 Java 代碼

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 控制台中，選擇 ka-app-code-*<username>* 桶，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 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 Apache Flink 應用程式的受管理服務

依照以下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 <https://console.aws.amazon.com/flink> 開啟適用於阿帕奇 Flink 的受管理服務
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**。
  - 對於執行期，選擇 Apache Flink。

#### Note

阿帕奇梁目前不與阿帕奇 Flink 版本 1.19 或更高版本兼容。

- 從版本下拉式清單中選取 Apache Flink 版本 1.15。
4. 對於 [存取權限]，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

當您使用主控台為 Apache Flink 應用程式建立受管理服務時，您可以選擇為應用程式建立IAM角色和原則。應用程式使用此角色和政策來存取其相依資源。這些IAM資源會使用您的應用程式名稱和區域命名，如下所示：

- 政策：`kinesis-analytics-service-MyApplication-us-west-2`
- 角色：`kinesis-analytics-MyApplication-us-west-2`

### 編輯IAM策略

編輯原IAM則以新增存取 Kinesis 資料串流的權限。

1. 在開啟IAM主控台<https://console.aws.amazon.com/iam/>。
2. 選擇政策。選擇主控台為您在上節所建立的 **kinesis-analytics-service-MyApplication-us-west-2** 政策。
3. 在摘要頁面，選擇編輯政策。選擇索引標籤。
4. 將下列政策範例的反白部分新增至政策。取代範例帳戶 IDs (**012345678901**) 使用您的帳戶 ID。

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

```

    "Resource": [
      "arn:aws:logs:us-west-2:012345678901:log-group:*"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
  }
]
}

```

## 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 **basic-beam-app-1.0.jar**。
3. 在 [存取應用程式資源] 下，針對 [存取權限] 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                  |
|---------------------------|-----------------|--------------------|
| BeamApplicationProperties | InputStreamName | ExampleInputStream |

| 群組 ID                            | 金鑰                      | 值                          |
|----------------------------------|-------------------------|----------------------------|
| <b>BeamApplicationProperties</b> | <b>OutputStreamName</b> | <b>ExampleOutputStream</b> |
| <b>BeamApplicationProperties</b> | <b>AwsRegion</b>        | <b>us-west-2</b>           |

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 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。

## 清理 AWS 資源

本節包括清除在「暫停視窗」教學課程中建立之 AWS 資源的程序。

本主題包含下列章節：

- [刪除適用於 Apache Flink 應用程式的受管理服務](#)
- [刪除您的 Kinesis 資料串流](#)

- [刪除您的 Amazon S3 對象和存儲桶](#)
- [刪除您的IAM資源](#)
- [刪除您的 CloudWatch 資源](#)

## 刪除適用於 Apache Flink 應用程式的受管理服務

1. 在 <https://console.aws.amazon.com/flink> 開啟適用於阿帕奇 Flink 的受管理服務
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，選擇刪除，然後確認刪除。

## 刪除您的 Kinesis 資料串流

1. 在 <https://console.aws.amazon.com/Kinesis> 處開啟室壁運動主控台。
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。

## 刪除您的 Amazon S3 對象和存儲桶

1. 在開啟 Amazon S3 主控台<https://console.aws.amazon.com/s3/>。
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。

## 刪除您的IAM資源

1. 在開啟IAM主控台<https://console.aws.amazon.com/iam/>。
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-we st-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 us-west-2 角色。

8. 選擇刪除角色，然後確認刪除。

## 刪除您的 CloudWatch 資源

1. 在開啟 CloudWatch 主控台<https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽列中，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

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

- [使用適用於 Apache Flink 的 Amazon 受管服務部署、操作和擴展應用程式](#)
- [在部署至 Apache Flink 的受管理服務之前，先在本機開發 Apache Flink 應用程式](#)
- [使用事件偵測與管理服務的 Apache Flink 工作室](#)
- [使用適用於 Amazon Kinesis 的 AWS 串流資料解決方案](#)
- [練習使用點擊流實驗室與阿帕奇 Flink 和阿帕奇卡夫卡](#)
- [使用 Application Auto Scaling 放設定自訂縮放](#)
- [查看示例 Amazon CloudWatch 儀表板](#)
- [使用適用於 Amazon 的 AWS 串流資料解決方案範本 MSK](#)
- [探索更多適用於 Apache Flink 解決方案的託管服務 GitHub](#)

## 使用適用於 Apache Flink 的 Amazon 受管服務部署、操作和擴展應用程式

本研討會涵蓋以 Java 開發 Apache Flink 應用程式、如何在您的應用程式中執行和除錯 IDE，以及在適用於 Apache Flink 的 Amazon 管理服務上進行封裝、部署和執行。您還將學習如何擴展，監控和故障排除應用程序。

[阿帕奇 Flink 研討會的 Amazon 託管服務。](#)

## 在部署至 Apache Flink 的受管理服務之前，先在本機開發 Apache Flink 應用程式

本研討會示範如何在本機上開始開發 Apache Flink 應用程式的基本知識，其長期目標是為 Apache Flink 部署至 Apache Flink 的受管理服務。

[使用阿帕奇 Flink 進行本地發展的入門指南](#)

## 使用事件偵測與管理服務的 Apache Flink 工作室

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。

### [事件檢測與阿帕奇 Flink 的管理服務](#)

## 使用適用於 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 管理服務選項提供 end-to-end 串流ETL範例，展示實際應用程式，該應用程式會在模擬的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：

- 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角色和策略。

### [Amazon Kinesis 動的串流資料解決方案](#)

## 練習使用點擊流實驗室與阿帕奇 Flink 和阿帕奇卡夫卡

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。

### [點擊流實驗室](#)

## 使用 Application Auto Scaling 放設定自訂縮放

兩個範例說明如何使用應用程式自動擴展功能，為 Apache Flink 應用程式自動調整管理服務的規模。這可讓您設定自訂資源調度政策和自訂縮放屬性。

- [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](#)。

## 查看示例 Amazon CloudWatch 儀表板

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。

[阿帕奇 Flink 指標儀表板的託管服務](#)

## 使用適用於 Amazon 的 AWS 串流資料解決方案範本 MSK

適用於 Amazon 的 AWS 串流資料解決方案 MSK 提供 AWS CloudFormation 範本，讓資料流經生產者、串流儲存、消費者和目的地。

[AWS Amazon 的流數據解決方案 MSK](#)

## 探索更多適用於 Apache Flink 解決方案的託管服務 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 — 基準測試公用程式](#)
- [快照管理器 — Amazon Managed Service for Apache Flink](#)
- [ETL 使用阿帕奇 Flink 和 Amazon 管理服務為阿帕奇 Flink 進行流式傳輸](#)
- [對客戶意見反饋進行即時情緒分析](#)

# 為 Apache Flink 管理服務使用實用的公用程式

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](#)
- [基準測試](#)

## 快照管理員

對於 Flink 應用程式來說，定期啟動儲存點/快照，以便進行更順暢的故障復原是最佳實務。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](#)

## 為 Apache Flink 應用程式建立及使用受管理服務的範例

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](#)
- [教學課程：開始使用適用於 Apache Flink 的 DataStream API 受管理服務](#)

### Note

這些範例假設您使用的是美國東部 (維吉尼亞北部) 區域 (us-east-1)。如果您使用不同的區域，請適當地更新應用程式程式碼、命令和IAM角色。

### 主題

- [適用於阿帕奇 Flink 管理服務的 Java 範例](#)
- [阿帕奇 Flink 管理服務的 Python 示例](#)
- [斯卡拉示例為阿帕奇 Flink 管理服務](#)

## 適用於阿帕奇 Flink 管理服務的 Java 範例

下面的實例演示如何創建用 Java 編寫的應用程序。

### Note

大多數範例都設計為在本機、您的開發機器和您IDE選擇的機器上執行，以及在 Apache Flink 的 Amazon 受管服務上執行。它們會示範您可以用來傳遞應用程式參數的機制，以及如何正確設定相依性，以便在兩個環境中執行應用程式，而不需要變更。

## 開始使用 DataStream API

此範例顯示一個簡單的應用程式，從 Kinesis 資料串流讀取並寫入另一個 Kinesis 資料串流，並使用 DataStream API 此範例示範如何使用正確的相依性設定檔案、建立 uber-JAR，然後剖析組態參數，以便您可以在本機IDE、您的和 Apache Flink 的 Amazon 受管服務上執行應用程式。

代碼示例：[GettingStarted](#)

## 開始使用桌子API和 SQL

此範例顯示使用表的簡單應用TableAPI程式SQL。它示範如何DataStreamAPI與TableAPI或SQL在相同的 Java 應用程式中整合。它還示範如何使用DataGen連接器從 Flink 應用程式本身產生隨機測試資料，而不需要外部資料產生器。

完整的例子：[GettingStartedTable](#)

## 使用水槽 () DataStream API

此範例示範如何使用 DataStream API's FileSink 將JSON檔案寫入 S3 儲存貯體。

[範例程式碼](#)：

## 使用 Kinesis 來源、標準或EFO消費者，以及接收器 () DataStream API

本範例示範如何使用標準使用者或設定 Kinesis 資料串流的來源消耗EFO，以及如何設定 Kinesis 資料串流的接收器。

代碼示例：[KinesisConnectors](#)

## 使用 Amazon 數據 Firehose 收器 ( ) DataStreamAPI

此範例說明如何將資料傳送至 Amazon 資料 Firehose (以前稱為 Kinesis Data Firehose)。

代碼示例：[KinesisFirehoseSink](#)

## 使用窗口聚合 ( ) DataStream API

此範例將示範中的四種視窗聚合類型。DataStream API

1. 基於處理時間的滑動窗口
2. 基於事件時間滑動窗口
3. 基於處理時間的翻滾窗口

## 4. 基於事件時間的翻滾窗口

程式碼範例：[視窗](#)

### 使用自訂指標

此範例顯示如何將自訂指標新增至 Flink 應用程式，並將其傳送至 CloudWatch 指標。

代碼示例：[CustomMetrics](#)

## 阿帕奇 Flink 管理服務的 Python 示例

以下實例演示瞭如何創建用 Python 編寫的應用程式。

### Note

大多數範例都設計為在本機、您的開發機器和您IDE選擇的機器上執行，以及在 Apache Flink 的 Amazon 受管服務上執行。它們示範可用來傳遞應用程式參數的簡單機制，以及如何正確設定相依性，以便在兩個環境中執行應用程式，而不需要變更。

### 專案相依性

大多數 PyFlink 範例都需要JAR檔案形式的一或多個相依性，例如 Flink 連接器。然後，在 Apache Flink 的 Amazon 受管服務上部署時，必須將這些相依性與應用程式一起封裝。

下列範例已包含可讓您在本機執行應用程式以進行開發和測試，以及正確封裝所需相依性的工具。這種工具需要使用 Java JDK11 和阿帕奇 Maven 的。如需特定指示，請參閱每個範例中README包含的。

### 範例

#### 開始使用 PyFlink

這個例子演示了使用SQL嵌入在 Python 代碼的 PyFlink 應用程式的基本結構。此專案也會為包含連接器等JAR相依性的任何 PyFlink 應用程式提供架構。本README節提供有關如何在本機執行 Python 應用程式進行開發的詳細指引。此範例也會示範如何在此範例中包含單一JAR相依性 (Kinesis SQL 連接器) 在您的 PyFlink應用程式中。

代碼示例：[GettingStarted](#)

## 使用窗口聚合 ( ) DataStream API

這個例子演示了四種類型的SQL嵌入在 Python 應用程序的窗口聚合。

1. 基於處理時間的滑動窗口
2. 基於事件時間滑動窗口
3. 基於處理時間的翻滾窗口
4. 基於事件時間的翻滾窗口

程式碼範例：[視窗](#)

## 使用 S3 水槽

這個範例說明如何使用SQL內嵌在 Python 應用程式中，將輸出當做JSON檔案寫入 Amazon S3。您必須啟用 S3 接收器的檢查點，才能將檔案寫入和旋轉至 Amazon S3。

範例程式碼：[視窗](#)

## 使用使用者定義函數 (UDF)

這個例子演示了如何定義用戶定義功能，在 Python 中實現它，並在 Python 應用程序中運行的SQL代碼中使用它。

代碼示例：[UDF](#)

## 使用 Amazon 數據 Firehose 收器

這個例子演示了如 Firehose 用SQL。

代碼示例：[FirehoseSink](#)

## 斯卡拉示例為阿帕奇 Flink 管理服務

下列範例示範如何搭配使用 Scala 與 Apache Flink 來建立應用程式。

### 設定多步驟應用程式

這個例子演示了如何在 Scala 中設置一個 Flink 應用程序。它演示瞭如何配置項SBT目以包含依賴關係並構建 uber-JAR。



代碼示例：[GettingStarted](#)

# Amazon Managed Service for Apache Flink 中的安全性

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您將受益於資料中心和網路架構，專為滿足最敏感安全性組織的需求而打造。

安全是 AWS 與您之間共同的責任。[共同責任模型](#) 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。AWS 還為您提供可以安全使用的服務。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](#)。
- 雲端中的安全性 — 您的責任取決於您使用的 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

- [適用於 Apache Flink 的 Amazon 受管服務中的資料保護](#)
- [Amazon Managed Service for Apache Flink 的身分和存取管理](#)
- [適用於 Apache Flink 的 Amazon 受管服務的合規驗證](#)
- [Amazon Managed Service for Apache Flink 中的彈性](#)
- [Apache Flink 受管理服務中的基礎架構安全性](#)
- [適用於 Apache Flink 的受管理服務的安全性最佳做法](#)

## 適用於 Apache Flink 的 Amazon 受管服務中的資料保護

您可以使用提供的工具保護您的數據 AWS。適用於 Apache Flink 的受管服務可與支援加密資料的服務搭配使用，包括 Firehose 和 Amazon S3。

## 阿帕奇 Flink 管理服務中的數據加密

### 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 資料串流的伺服器端加密？](#)。
- 輸出資料可以使用 Firehose 進行靜態加密，將資料存放在加密的 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，請參閱使用 [KMS—Managed 金鑰使用伺服器端加密保護資料 \(SSE-KMS\)](#)。
- 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。
- Apache Flink 和 Firehose 服務之間傳輸中的資料。

## 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) 可協助系統管理員安全地控制 AWS 資源存取權。AWS 服務 IAM系統管理員控制誰可以驗證 (登入) 和授權 (具有權限) 使用 Apache Flink 資源的受管理服務。IAM是您 AWS 服務 可以免費使用的。

## 主題

- [物件](#)
- [使用身分驗證](#)
- [使用政策管理存取權](#)
- [阿帕奇 Flink 的 Amazon 託管服務如何與 IAM](#)
- [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

根據您在 Apache Flink 的受管理服務中所做的工作而有所不同 AWS Identity and Access Management (IAM) 的使用方式。

**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，您可能需要額外的許可。了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。然後，您必須向IAM管理員提交請求，才能變更服務使用者的權限。檢閱此頁面上的資訊，以瞭解的基本概念IAM。若要深入瞭解貴公司如何搭配 Apache Flink IAM 的受管理服務搭配使用，請參閱[阿帕奇 Flink 的 Amazon 託管服務如何與 IAM](#)。

**IAM系統管理員 —**如果您是IAM系統管理員，您可能想要瞭解如何撰寫原則以管理 Apache Flink 受管理服務存取權的詳細資訊。若要檢視您可以在IAM中使用的 Apache Flink 身分型原則的受管理服務範例，請參閱。[Amazon Managed Service for Apache Flink 的身分型政策範例](#)

## 使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以IAM使用者身分或假設IAM角色來驗證 (登入 AWS)。AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。AWS IAM Identity Center (IAM身分識別中心) 使用者、貴公司的單一登入驗證，以及您的 Google 或 Facebook 認證都是

聯合身分識別的範例。當您以同盟身分登入時，您的管理員先前會使用IAM角色設定聯合身分識別。當您使 AWS 用同盟存取時，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。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》AWS 帳戶中的[如何登入](#)您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以密碼編譯方式簽署您的要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署要求的詳細資訊，請參閱使用IAM者指南中的[簽署 AWS API要求](#)。

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) 來增加帳戶的安全性。若要深入瞭解，請參閱使用AWS IAM Identity Center 者指南中的[多重要素驗證](#)和[使用多重要素驗證 \(MFA\) AWS的](#)使用IAM者指南。

## AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。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。如需需要您以 root 使用者身分登入的完整工作清單，請參閱《使用指南》中的[〈需要 root 使用者認證的IAM工作〉](#)。

## 聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時認證 AWS 服務 來存取。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identity Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

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，也可以連線並同步處理至您自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需IAM身分識別中心的相關資訊，請參閱[IAM識別中心是什麼？](#) 在《AWS IAM Identity Center 使用者指南》中。

## IAM 使用者和群組

[IAM使用者](#)是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定權限。在可能的情況下，我們建議您仰賴臨時登入資料，而不要建立具有長期認證 (例如密碼和存取金鑰) 的IAM使用者。不過，如果

您的特定使用案例需要使用IAM者的長期認證，建議您輪換存取金鑰。如需詳細資訊，請參閱《[使用指南](#)》中的「IAM定期輪換存取金鑰」以瞭解需要長期認證的使用案例。

[IAM群組](#)是指定IAM使用者集合的身分識別。您無法以群組身分簽署。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程序變得更為容易。例如，您可以擁有一個名為的群組，IAMAdmins並授與該群組管理IAM資源的權限。

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。要了解更多信息，請參閱《[IAM用戶指南](#)》中的[創建用戶（而不是角色）的IAM時間](#)。

## IAM角色

[IAM角色](#)是您 AWS 帳戶中具有特定權限的身份。它類似於用IAM戶，但不與特定人員相關聯。您可以 AWS Management Console 透過[切換角色來暫時擔任中的角色](#)。IAM您可以透過呼叫 AWS CLI 或 AWS API作業或使用自訂來擔任角色URL。如需有關使用角色方法的詳細資訊，請參閱《[使用指南](#)》中的[IAM〈使用IAM角色〉](#)。

IAM具有臨時認證的角色在下列情況下很有用：

- 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，並獲授予由角色定義的許可。如需聯合角色的相關資訊，請參閱《[使用指南](#)》中的[〈建立第三方身分識別提供IAM者的角色〉](#)。如果您使用IAM身分識別中心，則需要設定權限集。為了控制身分驗證後可以存取的內IAM容，IAMIdentity Center 會將權限集與中的角色相關聯。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使用者權限 — IAM 使用者或角色可以假定某個IAM角色，暫時取得特定工作的不同權限。
- 跨帳戶存取 — 您可以使用IAM角色允許不同帳戶中的某個人(受信任的主體)存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源(而不是使用角色作為代理)。若要瞭解跨帳戶存取角色與以資源為基礎的政策之間的差異，請參閱《[IAM使用指南](#)》[IAM中的〈跨帳號資源存取〉](#)。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中撥打電話時，該服務通常會在 Amazon 中執行應用程式EC2或將物件存放在 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。
- 轉寄存取工作階段 (FAS) — 當您使用IAM者或角色執行中的動作時 AWS，您會被視為主參與者。使用某些服務時，您可能執行某個動作，進而在不同服務中啟動另一個動作。FAS會使用主參與者呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。FAS只有當服務收到

需要與其他 AWS 服務 資源互動才能完成的請求時，才會發出請求。在此情況下，您必須具有執行這兩個動作的許可。有關提出FAS請求時的策略詳細信息，請參閱[轉發訪問會話](#)。

- 服務角色 — 服務角色是指服務代表您執行動作所代表的IAM角色。IAM管理員可以從中建立、修改和刪除服務角色IAM。如需詳細資訊，請參閱《IAM使用指南》AWS 服務中的[建立角色以將權限委派給](#)
- 服務連結角色 — 服務連結角色是連結至 AWS 服務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中，AWS 帳戶 且屬於服務所有。IAM管理員可以檢視 (但無法編輯服務連結角色) 的權限。
- 在 Amazon 上執行的應用程式 EC2 — 您可以使用IAM角色來管理在執行個體上EC2執行的應用程式以及發出 AWS CLI 或 AWS API請求的臨時登入資料。這比在EC2實例中存儲訪問密鑰更好。若要將 AWS 角色指派給EC2執行個體並讓其所有應用程式都能使用，請建立附加至執行個體的執行個體設定檔。執行個體設定檔包含角色，可讓執行個體上EC2執行的程式取得臨時登入資料。如需詳細資訊，請參閱[使用者指南中的使用IAM角色將許可授與在 Amazon EC2 執行個體上執行的應IAM用程式](#)。

要了解是否使用IAM角色還是用IAM戶，請參閱 [《用戶指南》中的「IAM創建IAM角色的時機 \(而不是用戶\)」](#)。

## 使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以JSON文件的形式儲存在中。如需有關JSON原則文件結構和內容的詳細資訊，請參閱《IAM使用指南》中的策略[概觀](#)。JSON

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。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。若要授與使用者對所需資源執行動作的權限，IAM管理員可以建立IAM策略。然後，系統管理員可以將IAM原則新增至角色，使用者可以擔任這些角色。

IAM原則會定義動作的權限，不論您用來執行作業的方法為何。例如，假設您有一個允許 iam:GetRole 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或取得角色資訊 AWS API。

## 身分型政策

以身分識別為基礎的原則是您可以附加至身分識別 (例如使用者、使用IAM者群組或角色) 的JSON權限原則文件。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。若要瞭解如何建立以身分識別為基礎的策略，請參閱《IAM使用指南》中的 [〈建立IAM策略〉](#)。

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。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶。受管政策包括 AWS 受管政策和客戶管理的策略。若要了解如何在受管策略或內嵌策略之間進行選擇，請參閱《IAM使用手冊》中的「[在受管策略和內嵌策略之間進行選擇](#)」。

## 資源型政策

以資源為基礎的JSON策略是您附加至資源的政策文件。以資源為基礎的政策範例包括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限。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](#)。主參與者可以包括帳戶、使用者、角色、同盟使用者或。AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的策略IAM中使用 AWS 受管政策。

## 存取控制清單 (ACLs)

存取控制清單 (ACLs) 控制哪些主參與者 (帳戶成員、使用者或角色) 具有存取資源的權限。ACLs類似於以資源為基礎的策略，雖然它們不使用JSON政策文件格式。

Amazon S3 和 Amazon VPC 是支持服務的示例ACLs。AWS WAF若要進一步了解ACLs，請參閱 Amazon 簡單儲存服務開發人員指南中的存取控制清單 [\(ACL\) 概觀](#)。

## 其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- **權限界限** — 權限界限是一項進階功能，您可以在其中設定以身分識別為基礎的原則可授與給IAM實體 (IAM使用者或角色) 的最大權限。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。如需有關權限界限的詳細資訊，請參閱《IAM使用指南》中的 [IAM實體的權限界限](#)。
- **服務控制策略 (SCPs)** — SCPs 是指定中組織或組織單位 (OU) 最大權限的JSON策略 AWS Organizations。AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個



服務。如果您啟用組織中的所有功能，則可以將服務控制策略 (SCPs) 套用至您的任何或所有帳戶。SCP限制成員帳戶中實體的權限，包括每個帳戶 AWS 帳戶根使用者。如需有關 Organizations 的詳細資訊 SCPs，請參閱 AWS Organizations 使用指南中的 [服務控制原則](#)。

- 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。如需詳細資訊，請參閱 IAM 使用指南中的 [工作階段原則](#)。

## 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。若要瞭解如何在涉及多個原則類型時 AWS 決定是否允許要求，請參閱 IAM 使用指南中的 [原則評估邏輯](#)。

## 阿帕奇 Flink 的 Amazon 託管服務如何與 IAM

在您用 IAM 來管理 Apache Flink 受管理服務的存取權限之前，請先瞭解哪些 IAM 功能可與 Apache Flink 的受管理服務搭配使用。

IAM 您可以與阿帕奇 Flink 的 Amazon 託管服務一起使用的功能

| IAM 特徵                       | Managed Service for Apache Flink 支援 |
|------------------------------|-------------------------------------|
| <a href="#">身分型政策</a>        | 是                                   |
| <a href="#">資源型政策</a>        | 否                                   |
| <a href="#">政策動作</a>         | 是                                   |
| <a href="#">政策資源</a>         | 是                                   |
| <a href="#">政策條件索引鍵</a>      | 否                                   |
| <a href="#">ACLs</a>         | 否                                   |
| <a href="#">ABAC(策略中的標籤)</a> | 是                                   |
| <a href="#">暫時性憑證</a>        | 是                                   |
| <a href="#">主體許可</a>         | 是                                   |

| IAM特徵                  | Managed Service for Apache Flink 支援 |
|------------------------|-------------------------------------|
| <a href="#">服務角色</a>   | 否                                   |
| <a href="#">服務連結角色</a> | 否                                   |

若要取得 Apache Flink 和其他 AWS 服務的受管理服務如何與大部分IAM功能搭配運作的高階檢視，請參閱《IAM使用者指南》IAM中的適用[AWS 服務](#)。

## Managed Service for Apache Flink 內的身分型政策

支援身分型政策：是

以身分識別為基礎的原則是您可以附加至身分識別 (例如使用者、使用IAM者群組或角色) 的JSON權限原則文件。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。若要瞭解如何建立以身分識別為基礎的策略，請參閱《IAM使用指南》中的〈[建立IAM策略](#)〉。

使用以IAM身分識別為基礎的策略，您可以指定允許或拒絕的動作和資源，以及允許或拒絕動作的條件。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。若要瞭解可在JSON策略中使用的所有元素，請參閱《使用IAM者指南》中的[IAMJSON策略元素參考資料](#)。

## 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

支援以資源為基礎的政策：是

以資源為基礎的JSON策略是您附加至資源的政策文件。以資源為基礎的政策範例包括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限。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](#)。主參與者可以包括帳戶、使用者、角色、同盟使用者或。AWS 服務

若要啟用跨帳戶存取，您可以在以資源為基礎的策略中指定一個或多個帳戶中的一個或多個帳戶中的IAM實體作為主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主參與者和資源不同時 AWS 帳戶，受信任帳戶中的IAM管理員也必須授與主參與者實體 (使用者或角色) 權限，才能存取資

源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM使用指南》[IAM中的〈跨帳號資源存取〉](#)。

## 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策略來指定誰可以存取什麼內容。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策略Action元素描述了您可以用來允許或拒絕策略中存取的動作。策略動作通常與關聯 AWS API操作具有相同的名稱。有一些例外情況，例如沒有匹配API操作的僅限權限的操作。也有一些作業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*"
```

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](#)。

## 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策略來指定誰可以存取什麼內容。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。

ResourceJSON原則元素會指定要套用動作的一或多個物件。陳述式必須包含 Resource 或 NotResource 元素。最佳做法是使用其 [Amazon 資源名稱 \(ARN\)](#) 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 (例如列出操作)，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 Apache Flink 資源類型及其受管服務的清單ARNs，請參閱服務授權參考資料中的 Amazon 針對 Apache Flink 受管服務 [定義](#) 的資源。若要了解您可以針對每個資源指定哪些動作，請參閱 [Amazon 受管服務為 Apache Flink 定義ARN](#) 的動作。

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](#)。

## 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金鑰：是

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。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。如果您為單一條件索引鍵指定多個值，請使用邏輯 OR 運算來 AWS 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，只有在 IAM 使用者名稱標記資源時，您才可以授與 IAM 使用者存取資源的權限。如需詳細資訊，請參閱《IAM 使用指南》中的 [IAM 政策元素：變數和標籤](#)。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《使用指南》中的[AWS 全域條件內IAM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](#)。

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](#)。

## 適用於 Apache Flink 的受管理服務中的存取控制清單 (ACLs)

支援ACLs：無

存取控制清單 (ACLs) 控制哪些主參與者 (帳戶成員、使用者或角色) 具有存取資源的權限。ACLs類似於以資源為基礎的策略，雖然它們不使用JSON政策文件格式。

## 基於屬性的訪問控制 ( ABAC ) 與 Apache Flink 的託管服務

支援 ABAC (策略中的標籤): 是

以屬性為基礎的存取控制 (ABAC) 是一種授權策略，可根據屬性定義權限。在中 AWS，這些屬性稱為標籤。您可以將標籤附加至IAM實體 (使用者或角色) 和許多 AWS 資源。標記實體和資源是的第一步 ABAC。然後，您可以設計ABAC策略，以便在主參與者的標籤與他們嘗試存取的資源上的標籤相符時允許作業。

ABAC在快速成長的環境中很有幫助，並且有助於原則管理變得繁瑣的情況。

如需根據標籤控制存取，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金鑰，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金鑰，則值為 Partial。

如需有關的詳細資訊ABAC，請參閱[什麼是ABAC？](#) 在《IAM使用者指南》中。若要檢視包含設定步驟的自學課程ABAC，請參閱《[使用指南](#)》中的〈[使用以屬性為基礎的存取控制 \(ABAC\) IAM](#)〉。

## 將臨時憑證用於 Managed Service for Apache Flink

支援臨時憑證：是

當您使用臨時憑據登錄時，某些 AWS 服務不起作用。如需其他資訊，包括哪些 AWS 服務與臨時登入資料搭配使用 [AWS 服務](#)，請參閱《IAM使用指南》IAM中的使用方式。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立臨時認證。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。如需有關切換角色的詳細資訊，請參閱《IAM使用者指南》中的 [〈切換到角色 \(主控台\)〉](#)。

您可以使用 AWS CLI 或手動建立臨時認證 AWS API。然後，您可以使用這些臨時登入資料來存取 AWS。AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細 [資訊](#)，請參閱IAM。

## 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

當您使用使用IAM者或角色在中執行動作時 AWS，您會被視為主參與者。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會使用主參與者呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。FAS只有當服務收到需要與其他 AWS 服務 資源互動才能完成的請求時，才會發出請求。在此情況下，您必須具有執行這兩個動作的許可。有關提出FAS請求時的策略詳細信息，請參閱[轉發訪問會話](#)。

## Managed Service for Apache Flink 的服務角色

支援服務角色：是

服務角色是服務假定代表您執行動作的[IAM角色](#)。IAM管理員可以從中建立、修改和刪除服務角色 IAM。如需詳細資訊，請參閱《IAM使用指南》AWS 服務中的[建立角色以將權限委派給](#)

### 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

服務連結角色是連結至 AWS 服務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中，AWS 帳戶 且屬於服務所有。IAM管理員可以檢視 (但無法編輯服務連結角色) 的權限。

如需有關建立或管理服務連結角色的詳細資訊，請參閱[使用IAM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) 或執行工作 AWS API。若要授與使用者對所需資源執行動作的權限，IAM管理員可以建立IAM策略。然後，系統管理員可以將IAM原則新增至角色，使用者可以擔任這些角色。

若要瞭解如何使用這些範例原則文件來建立以IAM身分識別為基礎的JSON策略，請參閱使用指南中的[IAM建立IAM策略](#)。

有關 Apache Flink 的受管服務定義的動作和資源類型的詳細資訊，包括每種資源類型的格式，請參閱服務授權參考中的 [Apache Flink 適用於 Amazon 受管服務的動作、資源和條件金鑰](#)。ARNs

### 主題

- [政策最佳實務](#)
- [使用 Managed Service for Apache Flink 主控台](#)
- [允許使用者檢視他們自己的許可](#)

## 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 受管原則並邁向最低權限權限 — 若要開始將權限授與使用者和工作負載，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們在您的 AWS 帳戶。建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。[如需詳細資訊，請參閱AWS 《IAM使用指南》中針對工作職能的AWS 受管理策略或受管理的策略。](#)
- 套用最低權限權限 — 當您使用原則設定權限時，IAM只授與執行工作所需的權限。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。如需有關使用套用權限IAM的詳細資訊，請參閱《使用指南》[IAM中的IAM 《策略與權限》](#)。
- 使用IAM策略中的條件進一步限制存取 — 您可以在策略中新增條件，以限制對動作和資源的存取。例如，您可以撰寫政策條件，以指定必須使用傳送所有要求SSL。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱《IAM使用指南》中的[IAMJSON策略元素：條件](#)。

- 使用 IAM Access Analyzer 驗證您的原IAM則，以確保安全性和功能性的權限 — IAM Access Analyzer 會驗證新的和現有的原則，以便原則遵循IAM原則語言 (JSON) 和IAM最佳做法。IAMAccess Analyzer 提供超過 100 項原則檢查和可行的建議，協助您撰寫安全且功能正常的原則。如需詳細資訊，請參閱[IAM使用指南中的存取分析器原則驗證](#)。
- 需要多因素驗證 (MFA) — 如果您的案例需要使IAM用者或 root 使用者 AWS 帳戶，請開啟以取得額外MFA的安全性。若要在呼叫API作業MFA時需要，請在原則中新增MFA條件。如需詳細資訊，請參閱《IAM使用指南》中的 [< 設定MFA受保護的API存取 >](#)。

如需有關中最佳作法的詳細資訊IAM，請參閱《IAM使用指南》IAM中的「[安全性最佳作法](#)」。

## 使用 Managed Service for 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。

您不需要為只對 AWS CLI 或撥打電話的使用者允許最低主控台權限 AWS API。相反地，只允許存取符合他們嘗試執行之API作業的動作。

若要確保使用者和角色仍可使用 Apache Flink 的受管理服務主控台，請同時將 Apache Flink 的受管理服務ConsoleAccess或ReadOnly AWS 受管理的原則附加至實體。如需詳細資訊，請參閱《[使用指南](#)》中的〈[將權限新增至IAM使用者](#)〉。

## 允許使用者檢視他們自己的許可

此範例顯示如何建立原則，讓使IAM用者檢視附加至其使用者身分識別的內嵌和受管理原則。此原則包含在主控台上或以程式設計方式使用或完成此動作的 AWS CLI 權限 AWS API。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "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

使用下列資訊可協助您診斷及修正使用 Apache Flink 和的受管理服務時可能會遇到的常見問題。IAM

### 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](#)
- [我想要允許我 AWS 帳戶以外的人員存取我的 Apache Flink 管理服務資源](#)

### 我未獲授權，無法在 Managed Service for Apache Flink 中執行動作

如果 AWS Management Console 告訴您您沒有執行動作的授權，則您必須聯絡您的管理員以尋求協助。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 允許您將現有角色傳遞給該服務，而不是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。

當名為的使用IAM者marymajor嘗試使用主控台在 Apache Flink 的受管理服務中執行動作時，就會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。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。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

## 我想要允許我 AWS 帳戶以外的人員存取我的 Apache Flink 管理服務資源

您可以建立一個角色，讓其他帳戶中的使用者或您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。對於支援以資源為基礎的政策或存取控制清單 (ACLs) 的服務，您可以使用這些政策授與人員存取您的資源。

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 [阿帕奇 Flink 的 Amazon 託管服務如何與 IAM](#)。
- 若要瞭解如何提供您所擁有資 AWS 帳戶 源的存取權，請參閱《[IAM使用指南](#)》中的〈[提供存取權給您 AWS 帳戶 所擁有的其他IAM使用者](#)〉。
- 若要瞭解如何將您的資源存取權提供給第三方 AWS 帳戶，請參閱《[IAM使用指南](#)》中的[提供第三方 AWS 帳戶 擁有的存取權](#)。

- 若要瞭解如何透過聯合身分識別提供存取權，請參閱[使用指南中的提供對外部驗證使用IAM者的存取權 \(身分聯合\)](#)。
- 若要瞭解針對跨帳號存取使用角色與以資源為基礎的政策之間的差異，請參閱《[使用IAM者指南](#)》IAM中的〈[跨帳號資源存取](#)〉。

## 預防跨服務混淆代理人

在中 AWS，當某個服務 (呼叫服務) 呼叫另一個服務 (呼叫的服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來對其他客戶的資源採取操作，即使該服務不應有適當的許可，導致混淆代理人的問題。

為了防止代表混淆，AWS 提供了一些工具，可協助您使用已授予您帳戶資源存取權的服務主體來保護所有服務的資料。本節重點介紹 Apache Flink 託管服務特定的跨服務混淆副預防，但是，您可以在IAM用戶指南的[混淆副問題部分中](#)了解有關此主題的更多信息。

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`。

的值 `aws:SourceArn` 必須是 Apache Flink ARN 的受管理服務所使用的資源，該資源的值必須是以下列格式指定：`arn:aws:kinesisanalytics:region:account:resource`

解決混淆的副問題的建議方法是將 `aws:SourceArn` 全局條件上下文鍵與完整資源一起使用 ARN。

如果您不知道資源 ARN 的完整內容，或者您要指定多個資源，請針對的未知部分使用萬用字元 (\*) 的 `aws:SourceArn` 金鑰 ARN。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：

如要防範混淆代理人問題

1. 登入 AWS 管理主控台，然後在的開啟 IAM 主控台 <https://console.aws.amazon.com/iam/>。
2. 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，將預設JSON原則取代為使用一個或兩個aws:SourceArn和aws:SourceAccount全域條件內容金鑰的原則。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"
        }
      }
    }
  ]
}
```

## 適用於 Apache Flink 的 Amazon 受管服務的合規驗證

第三方稽核員會在多個合規計劃中，評估適用於 Apache Flink 的 Amazon 受管服務的安全性和合 AWS 規性。這些包括SOCPCI、HIPAA、和其他。

如需特定規範遵循方案範圍內的 AWS 服務清單，請參閱。如需一般資訊，請參閱 [AWS 合規計劃](#)。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱 [下載中的報告 AWS Artifact](#)。

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。如果您使用 Apache Flink 受管理服務時，必須遵守HIPAA或等標準PCI，AWS 提供資源以協助：

- [安全性與合規性快速入門指南](#) — 這些部署指南討論架構考量，並提供在上部署以安全性和法規遵循為重點的基準環境的步驟。AWS

- 在 [Amazon Web Services 上進行HIPAA安全與合規架構](#)。本白皮書說明公司如何使用建立HIPAA符合標準的 AWS 應用程式。
- [AWS 法規遵循資源](#) — 此工作簿和指南集合可能適用於您的產業和所在地。
- [AWS Config](#)— 此 AWS 服務評估您的資源配置是否符合內部實踐，行業準則和法規。
- [AWS Security Hub](#)— 此 AWS 服務提供安全性狀態的全面檢視，協助 AWS 您檢查您是否符合安全性產業標準和最佳做法。

## 美聯儲 RAMP

AWS 美聯儲RAMP合規計劃包括 Apache Flink 作為美聯儲RAMP授權服務的託管服務。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。

[您可以通過 AWS 美聯儲 RAMPPMO，您的 AWS 銷售客戶經理請求訪問 Fed RAMP 安全包，或者您可以通過 AWS Artifact 下載它們。AWS](#)

如需詳細資訊，請參閱 [Fed RAMP](#)。

## Amazon Managed Service for Apache Flink 中的彈性

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。AWS 區域提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。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。

如需區域和可用區域的相關 AWS 資訊，請參閱[AWS 全域基礎結構](#)。

除了 AWS 全球基礎架構之外，Apache Flink 的受管服務還提供多種功能，協助支援您的資料恢復能力和備份需求。

## 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 JobManager 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。阿帕奇 Flink 的託管服務使用 Amazon 部署阿帕奇 Flink。EKSAmazon 中EKS針對跨可用區域

的每個 AWS 區域使用多個 Kubernetes 網繭。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，請參閱在 [Apache Flink 的受管理服務中實作容錯能力](#)。

## 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。

## Apache Flink 受管理服務中的基礎架構安全性

作為受管服務，Apache Flink 的受管服務受到 [Amazon Web Services：安 AWS 全流程概觀白皮書中所述的全球網路安全程序的](#)保護。

您可以使用 AWS 已發佈的 API 呼叫，透過網路存取 Apache Flink 的受管理服務。對 Apache Flink 的託管服務的所有 API 調用都通過傳輸層安全性 (TLS) 進行保護，並通過 IAM。用戶端必須支援 TLS 1.2 或更新版本。客戶還必須支持具有完美正向保密 ( ) 的密碼套件，例如短暫的迪菲-赫爾曼 (PFS) 或橢圓曲線短暫迪菲-赫爾曼 ( )。DHE 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 主體相關聯的秘密存取金鑰來簽署。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。

## 適用於 Apache Flink 的受管理服務的安全性最佳做法

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 Apache Flink 受管服務應用程式必須具有有效的登入資料，才能存取其他服務中的資源，例如 Kinesis 資料串流、Firehose 串流或 Amazon S3 儲存貯體。您不應將 AWS 登入資料直接存放在應用程式或 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。

相反地，您應該使用IAM角色來管理應用程式的臨時認證，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使用指南》中的下列主題：

- [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。您應該在相依資源中實作伺服器端加密，例如 Kinesis 資料串流、Firehose 串流和 Amazon S3 儲存貯體。如需在相依資源中實作伺服器端加密的詳細資訊，請參閱 [適用於 Apache Flink 的受管理服務中的資料保護](#)。

## 用 CloudTrail 於監控API通話

Apache Flink 的受管服務與服務整合在一起 AWS CloudTrail，該服務可提供 Apache Flink 受管服務中使用者、角色或 Amazon 服務所採取的動作記錄。

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 “日誌管理服務的阿帕奇 Flink API 呼叫 AWS CloudTrail”](#)。

# 在 Amazon 管理服務中為 Apache Flink 進行日誌記錄和監控

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。您應該從 AWS 解決方案的所有部分收集監視資料，以便在發生多點失敗時更輕鬆地偵錯。

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

- [登入管理服務](#)
- [在阿帕奇 Flink 的受管理服務中監視](#)
- [在 Apache Flink 的受管理服務中設定應用程式登入](#)
- [使用日誌見解分析 CloudWatch 日誌](#)
- [適用於 Apache Flink 的受管理服務中的度量和維度](#)
- [將自訂訊息寫入 CloudWatch 記錄](#)
- [日誌管理服務的阿帕奇 Flink API 呼叫 AWS CloudTrail](#)

## 登入管理服務

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 應用程式 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。

## 查詢日誌記錄與 CloudWatch 日誌見解

CloudWatch 日誌深入解析是大規模查詢記錄的強大服務。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](#)。

## 在阿帕奇 Flink 的受管理服務中監視

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[搭配 Amazon 管理服務搭配 Apache Flink 使用 CloudWatch 警示](#)。這些指標包括：

- `records_lag_max` 和 `millisBehindLatest`— 如果應用程式從 Kinesis 或 Kafka 取用，這些度量會指出應用程式是否落後，需要進行調整以跟上目前的負載。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。
- `cpuUtilization` 和 `heapMemoryUtilization`— 這些指標可以很好地指出應用程式的整體資源使用率，並且除非應用程式為 I/O 繫結，否則可用於主動擴充。
- `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。
- `numberOfFailedCheckpoints` 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)` 以發出梯度警示，而不是絕對值。

## 在 Apache Flink 的受管理服務中設定應用程式登入

透過將 Amazon CloudWatch 日誌記錄選項新增至 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 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 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。
- 當您使用新增 CloudWatch 記錄選項時 API，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串流所需的權限。

## 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* 是您的應用程式的名稱。

```
/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。如需應用程式日誌層級的相關資訊，請參閱[the section called “控制應用程式監控層”](#)。

## 使用設定 CloudWatch 記錄 CLI

若要使用新增 CloudWatch 記錄選項 AWS CLI，請完成下列操作：

- 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。

## 建立 CloudWatch 記錄群組和記錄串流

您可 CloudWatch 以使用記 CloudWatch 錄主控台或API。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動作可將 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。若要取得有關如何使用JSON檔案進API行動作輸入的資訊，請參閱 [〈〉適用於阿帕奇 Flink API 範例程式碼的託管服務](#)。

## 創建應用程序時添加 CloudWatch 日誌選項

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。在範例中，取代 *Amazon Resource Name (ARN) of the CloudWatch Log stream to add to the new application* 使用您自己的信息。如需該動作的詳細資訊，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 日誌選項添加到現有的應用程序

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。在範例中，取代每個 *user input placeholder* 使用您自己的信息。如需該動作的詳細資訊，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。在範例中，取代每個 *user input placeholder* 使用您自己的信息。如需該動作的詳細資訊，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

下面的實例演示瞭如何使用該 DeleteApplicationCloudWatchLoggingOption 操作來刪除現有的 CloudWatch 日誌選項。在範例中，取代每個 *user input placeholder* 使用您自己的信息。如需該動作的詳細資訊，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。

如需應用程式日誌層級的相關資訊，請參閱 [the section called “控制應用程式監控層”](#)。

## 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 您可以將這些權限新增至 Apache Flink 的受管理服務所承擔的 AWS Identity and Access Management (IAM) 角色。

如需將 IAM 角色用於 Apache Flink 的受管理服務的詳細資訊，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 Apache Flink 的受管理服務權限以擔任 IAM 角色，您可以將下列信任原則附加至服務執行角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

### 許可政策

若要授與 CloudWatch 從 Apache Flink 資源的受管理服務寫入記錄事件的應用程式權限，您可以使用下列 IAM 權限原則。為您的日誌群組和串流提供正確的 Amazon 資源名稱 (ARNs)。

```
{
  "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:*",
    ]
  }
]
```

## 控制應用程式監控層

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 “在 Apache Flink 的受管理服務中實作應用程式調整”](#)。
- 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 “在 Apache Flink 的受管理服務中使用運算子轉換資料”](#)。
- 平行處理層級：指標的適用範圍是應用程式平行處理層級。您只能使用的 [MonitoringConfigurationUpdate](#) 參數來設定此測量結果層次 [UpdateApplicationAPI](#)。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 “在 Apache Flink 的受管理服務中實作應用程式調整”](#)。

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。

## 套用記錄最佳做法

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。

## 執行記錄疑難排

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角色和原則是否正確。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 [???](#)。

## 使用 CloudWatch 日誌見解

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 “使用日誌見解分析 CloudWatch 日誌”](#)。

## 使用日誌見解分析 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 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 Route 53 或 Amazon 之類的服務 VPC，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。

### 執行 CloudWatch 日誌見解範例查詢

1. 在開啟 CloudWatch 主控台 <https://console.aws.amazon.com/cloudwatch/>。
2. 在導覽窗格中，選擇 Insights。
3. 螢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の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](#)。

## 檢閱範例查詢

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**

更換區域 (*us-west-2*)、帳戶識別碼 (*012345678901*) 和應用程式名稱 (*YourApplication*) 在下列查詢範例中顯示您應用程式的地區和您的帳戶 ID。

本主題包含下列章節：

- [分析操作：任務的分佈](#)
- [分析作業：平行處理原則的變更](#)
- [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：應用程式工作相關失敗](#)

## 分析操作：任務的分佈

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，請參閱在 [Apache Flink 的受管理服務中實作應用程式調整](#)。

```
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。如需設定運算子平行處理的詳細資訊，請參閱 [Apache Flink 文件](#) 中的 [設定平行處理：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
```

如需任務排程的詳細資訊，請參閱 [Apache Flink 文件](#) 中的 [作業和排程](#)。

## 分析作業：平行處理原則的變更

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 “在適用於 Apache Flink 的受管理服務中使用自動調整”](#)。

```
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:kinesisanalyticsus-
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:kinesisanalyticsus-
west-2:012345678901:application\YourApplication/
| filter @message like /ResourceNotFoundException/
| sort @timestamp desc
```

## 分析錯誤：應用程式工作相關失敗

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:kinesisanalyticsus-
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:kinesisanalyticsus-
west-2:012345678901:application\YourApplication/
| filter @message like /switched from RUNNING to FAILED/
| sort @timestamp desc
```

## 適用於 Apache Flink 的受管理服務中的度量和維度

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 | <p>* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。</p> <p>這些指標可用於識別應用程式中的瓶頸。</p> |
| 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 | 任務、運算子、平行處理層級 | <p>* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。</p>                           |

| 指標             | 單位  | 描述  | Level | 使用須知   |
|----------------|-----|---|-------|--|
|                |     | 值，則可以是 NaN。   |       | 這些指標可用於識別應用程式中的瓶頸。   |
| cpuUtilization | 百分比 | 工作管理員的整體CPU使用率百分比。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視應用程式中的最小、平均和最大使用CPU率。該CPUUtilization 指標僅考慮在容器內運行的TaskManager JVM進程的CPU使用情況。 |

| 指標                       | 單位  | 描述  | Level | 使用須知  |
|--------------------------|-----|---|-------|---|
| container CPUUtilization | 百分比 | Flink 應CPU 應用程式叢集中任務管理器容器的整體使用率百分比。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  | <p>它按每個容器計算如下：</p> <p>容器使用的總CPU時間 (秒) * 100/容器 CPU限制 (單位 CPUs/秒)</p> <p>該CPUUtilization 指標僅考慮在容器內運行的 TaskManager JVM進程的CPU使用情況。還有其他組件在同一個容器JVM內運行外部。該 container CPUUtilization 指標為您提供了更完整的圖片，包括容器CPU耗盡方面的所有過程以及由此引起的失敗。</p> |

| 指標                             | 單位  | 描述   | Level | 使用須知  |
|--------------------------------|-----|--|-------|---|
| container<br>MemoryUtilization | 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  | <p>它按每個容器計算如下：</p> <p>容器記憶體用量 (位元組) * 100 / 每 Pod 部署規格的容器記憶體限制 (位元組)</p> <p>HeapMemoryUtilization 和 ManagedMemoryUtilizations 標僅考慮特定的內存指標，如堆內存使用情況 TaskManager JVM 或託管內存 (像 <a href="#">RockSDB 狀態後端</a> 本機進程外 JVM 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，這是一個更佳的記憶體總量</p> |



| 指標                        | 單位  | 描述   | Level              | 使用須知  |
|---------------------------|-----|--|--------------------|---|
|                           |     |  |                    | 耗盡追蹤器。<br>一旦耗盡，它將導致該Out of Memory Error網 TaskManager 繭。                                   |
| container DiskUtilization | 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：<br><br>磁碟使用量 (位元組) * 100 / 容器的磁碟限制 (位元組)<br><br>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、任務、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。特定記憶體指標的 heapMemoryUtilization 唯一帳戶，例如 TaskManager JVM。 |
| 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。<br><br>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   |
|--------------------------------|-----|-----------------|-------|--|
| lastCheck<br>pointSize         | 位元組 | 最後一個檢查點的大小總計    | 應用程式  | <p>您可以使用此指標判斷執行中應用程式的儲存體使用率。</p> <p>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。</p>              |
| lastCheck<br>pointDura<br>tion | 毫秒  | 完成最後一個檢查點所花費的時間 | 應用程式  | <p>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，可能表示應用程式發生了問題，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。</p> |

| 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 | <p>* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。</p> <p>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。</p> |

| 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、任務、平行處理層級 | <p>* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。</p> <p>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。該指 ManagedMemoryUtilizations 標僅考慮特定的內存指標，例如託管內存（像 <a href="#">RockSDB 狀態後端JVM的本機進程外的內存使用情況</a>）</p> |

| 指標                        | 單位  | 描述  | Level                  | 使用須知   |
|---------------------------|-----|---|------------------------|--|
| managedMemoryUtilization* | 百分比 | 派生者<br>managedMemoryUsed/<br>managedMemoryTotal | 應用程式、運算子、任務、<br>平行處理層級 | <p>* 僅可用於執行 Flink 1.13 版本之 Managed Service for Apache Flink 應用程式。</p> <p>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。</p> |
| 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、任務、平行處理層級 | <p>* 若要套用一段時間內的SUM統計資料 (秒/分鐘)：</p> <ul style="list-style-type: none"> <li>• 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。</li> <li>• 由於 Apache Flink 的受管理服務每分鐘需要 4 個量度快照，因此應使用下列量度數學運算：<math>m1/4</math>，其中 <math>m1</math> 是一段時間內的SUM統計資料 (秒/分鐘)</li> </ul> <p>指標的「層級」指定此指標是衡量整個應用程式、特定運算子還是特定任務接收的記錄總數。</p> |



| 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、任務、平行處理層級 | <p>* 若要套用一段時間內的SUM統計資料 (秒/分鐘)：</p> <ul style="list-style-type: none"> <li>• 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。</li> <li>• 由於 Apache Flink 的受管理服務每分鐘需要 4 個量度快照，因此應使用下列量度數學運算：<math>m1/4</math>，其中 <math>m1</math> 是一段時間內的SUM統計資料 (秒/分鐘)</li> </ul> <p>指標的「層級」指定此指標是衡量整個應用程式、特定運算子還是特定任務每秒</p> |

| 指標 | 單位 | 描述 | Level | 使用須知     |
|----|----|----|-------|----------|
|    |    |    |       | 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、任務、平行處理層級 | <p>* 若要套用一段時間內的SUM統計資料 (秒/分鐘)：</p> <ul style="list-style-type: none"> <li>• 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。</li> <li>• 由於 Apache Flink 的受管理服務每分鐘需要 4 個量度快照，因此應使用下列量度數學運算：<math>m1/4</math>，其中 <math>m1</math> 是一段時間內的SUM統計資料 (秒/分鐘)</li> </ul> <p>指標的「層級」指定此指標是衡量整個應用程式、特定運算子還是特定任務發出的記錄總數。</p> |

| 指標                     | 單位 | 描述                 | Level | 使用須知  |
|------------------------|----|--------------------|-------|---|
| numLateRecordsDropped* | 計數 | 應用程式、運算子、任務、平行處理層級 |       | <p>* 若要套用一段時間內的SUM統計資料 (秒/分鐘)：</p> <ul style="list-style-type: none"> <li>• 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。</li> <li>• 由於 Apache Flink 的受管理服務每分鐘需要 4 個量度快照，因此應使用下列量度數學運算：<math>m1/4</math>，其中 <math>m1</math> 是一段時間內的SUM統計資料 (秒/分鐘)</li> </ul> <p>此運算子或任務因遲到而丟棄的記錄數。</p> |

| 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、任務、平行處理層級 | <p>* 若要套用一段時間內的SUM統計資料 (秒/分鐘)：</p> <ul style="list-style-type: none"> <li>• 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。</li> <li>• 由於 Apache Flink 的受管理服務每分鐘需要 4 個量度快照，因此應使用下列量度數學運算：<math>m1/4</math>，其中 <math>m1</math> 是一段時間內的SUM統計資料 (秒/分鐘)</li> </ul> <p>指標的「層級」指定此指標是衡量整個應用程式、特定運算子還是特定任務每秒</p> |

| 指標                   | 單位 | 描述                   | Level | 使用須知                                   |
|----------------------|----|----------------------|-------|--|
|                      |    |                      |       | 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。 |

| 指標     | 單位 | 描述              | Level | 使用須知   |
|--------|----|-----------------|-------|--|
| KPUs * | 計數 | 應用程式KPUs 使用的總數。 | 應用程式  | <p>* 此指標會在每個計費週期 (一小時) 接收一個樣本。若要視覺化KPUs 隨時間推移的數量，請使用 MAX或AVG超過至少一 (1) 小時的期間。</p> <p>計KPU數包括 orchestration KPU. 如需詳細資訊，請參閱 <a href="#">Apache Flink 定價的受管理服務</a>。</p> |

## Kinesis Data Streams 連接器測量結果

AWS 除了下列項目之外，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) | <ul style="list-style-type: none"> <li>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。</li> </ul> |

| 指標                     | 單位  | 描述                        | Level                       | 使用須知   |
|------------------------|-----|---------------------------|-----------------------------|--|
|                        |     |                           |                             | <ul style="list-style-type: none"> <li>值 -1 表示服務尚未報告指標的值。</li> </ul> |
| 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

AWS 除了以下內容之外，還會發MSK出 Amazon 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，如果啟用了位移遞交和檢查點。               | 應用程式、運算子、任務、平行處理層級                | 將位移遞交回 Kafka 只是公開取用者進度的一種手段，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、任務、平行處理層級                |  |



| 指標                  | 單位  | 描述   | Level                             | 使用須知 |
|---------------------|-----|--|-----------------------------------|------|
| committed offsets   | 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、任務、平行處理層級                |      |
| 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、任務、平行處理層級                |      |

## 阿帕奇齊柏林飛艇指标

對於 Studio 筆記本，AWS 會在應用程式層級發出下列指標：KPUcpuUtilization、heapMemoryUtilization、oldGenerationGCTime、oldGenerationGCTime 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 | 在 Apache 齊柏林飛艇服務器 CPU 使用率的總體百分比。   | 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 |

| 指標                   | 單位 | 描述                                 | Prometheus 名稱          |
|----------------------|----|------------------------------------|------------------------|
| 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 指標

您可以使用 CloudWatch Amazon 主 CloudWatch 控制台或 AWS CLI。

使用 CloudWatch 主控台檢視指標

1. 在開啟 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

- 在命令提示中，使用下列命令。

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

## 設定 CloudWatch 量度報告層級

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 Application 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。您只能使用設定此測量結果層次CLI。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](#)動作的範例要求會將 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](#)。

## 搭配適用於 Apache Flink 的 Amazon 受管服務使用自訂指標

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](#)
- [檢視建立對應類別的範例](#)
- [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 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「長條圖」Apache 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 Apache Flink 指標的詳細資訊，請參閱 [Apache Flink 文件](#) 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](#)

## 檢視建立對應類別的範例

下列程式碼範例示範如何建立可建立並遞增自訂量度的對應類別，以及如何透過將對應類別新增至 DataStream 物件，在應用程式中實作對應類別。

### 記錄計數自定義指標

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](#)。

## 字數自訂指標

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。

## 搭配 Amazon 管理服務搭配 Apache Flink 使用 CloudWatch 警示

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或多個動作。動作的範例是傳送通知給 Amazon 簡單通知服務 (AmazonSNS) 主題。

如需有關 CloudWatch 警示的詳細資訊，請參閱[使用 Amazon CloudWatch 警示](#)。

## 檢閱建議的鬧鐘

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用來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  |
|--|------|-----------|---|
| <code>downtime &gt; 0</code>                         | 平均數  | 0         | 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。建議所有應用程式使用。該Downtime指標測量中斷的持續時間。停機時間大於零表示應用程式失敗。如需疑難排解，請參閱 <a href="#">應用程式重新啟</a> 。   |
| <code>RATE (numberOfFailedCheckpoints) &gt; 0</code> | 平均數  | 0         | 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 (numberOfFailed檢查點) 以在梯度上發出警報，而不是絕對值。建議所有應用程式使用。使用此測量結果來監視應用程式健全狀況和檢查點進度。應用程式會 |



| 指標表達式  | 統計數字 | Threshold            | 描述   |
|--|------|----------------------|--|
| Operator.<br>numRecord<br>sOutPerSecond <<br>臨界值 | 平均數  | 在正常情況下，應用程式發出的記錄數下限。 | 在狀態良好時將狀態資料儲存至檢查點。如果應用程式在處理輸入資料時沒有取得進展，檢查點可能會因逾時而失敗。如需疑難排解，請參閱 <a href="#">檢查點逾時</a> 。<br><br>建議所有應用程式使用。低於此閾值可能表示應用程式未對輸入資料進行預期的進度。如需疑難排解，請參閱 <a href="#">輸送量太慢</a> 。 |

| 指標表達式  | 統計數字 | Threshold     | 描述  |
|--|------|---------------|---|
| <code>records_lag_max   millisbehindLatest &gt;</code> 臨界值 | 最大   | 正常情況下的最大預期延遲。 | 如果應用程式從 Kinesis 或 Kafka 取用，這些指標會指出應用程式是否落後，需要進行調整以跟上目前的負載。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。建議所有應用程式使用。將 <code>records_lag_max</code> 度量用於 Kafka 來源，或針對 <code>millisbehindLatest</code> 對 Kinesis 串流來源使用。超過此閾值可能表明應用程序沒有對輸入數據進行預期的進度。如需疑難排解，請參閱 <a href="#">輸送量太慢</a> 。 |

| 指標表達式                               | 統計數字 | Threshold              | 描述  |
|-------------------------------------|------|------------------------|---|
| lastCheck<br>pointDuration ><br>臨界值 | 最大   | 正常情況下的最大預期<br>檢查點持續時間。 | 監視狀態中儲存的資<br>料量，以及需要多長<br>時間才能取得檢查點<br>。如果檢查點增長或<br>花費較長時間，應用<br>程式會持續花費時間<br>在檢查點上，而且實際<br>處理的週期較少。在某<br>些時候，檢查點可能會<br>變得太大或花費很長時<br>間才會失敗。除了監控<br>絕對值之外，客戶還應<br>考慮使用 RATE(last<br>Checkpoin<br>tSize) 和<br>RATE(last<br>Checkpoin<br>tDuration)<br>監控變動率。如<br>果lastCheck<br>pointDuration 持<br>續增加，則超過此閾值<br>可能表示應用程序沒有<br>在輸入數據上進行預期的<br>進度，或者存在應用<br>程序健康問題（例如背<br>壓）。如需疑難排解，<br>請參閱 <a href="#">无界状态增长</a> 。 |

| 指標表達式                            | 統計數字 | Threshold            | 描述  |
|----------------------------------|------|----------------------|---|
| lastCheck<br>pointSize > 臨界<br>值 | 最大   | 正常情況下的最大預期<br>檢查點大小。 | 監視狀態中儲存的資<br>料量，以及需要多長<br>時間才能取得檢查點<br>。如果檢查點增長或<br>花費較長時間，應用<br>程式會持續花費時間<br>在檢查點上，而且實際<br>處理的週期較少。在某<br>些時候，檢查點可能會<br>變得太大或花費很長時<br>間才會失敗。除了監控<br>絕對值之外，客戶還應<br>考慮使用 RATE(last<br>Checkpoint<br>tSize) 和<br>RATE(last<br>Checkpoint<br>tDuration)<br>監控變動率。如<br>果lastCheck<br>pointSize 持續增<br>加，超過此閾值可能表<br>示應用程式正在累積狀<br>態資料。如果狀態資料<br>變得太大，從檢查點復<br>原時，應用程式可能會<br>耗盡記憶體，或者從檢<br>查點復原可能需要太長<br>時間。如需疑難排解，<br>請參閱 <a href="#">无界状态增长</a> 。 |

| 指標表達式                       | 統計數字 | Threshold  | 描述  |
|-----------------------------|------|--|---|
| heapMemoryUtilization > 臨界值 | 最大   | 這可以很好地指示應用程式的整體資源使用率，並且除非應用程式是 I/O 繫結，否則可用於主動擴充。正常情況下的最heapMemoryUtilization 大預期大小，建議值為 90%。 | 您可以使用此指標來監視整個應用程式中工作管理員的最大記憶體使用率。如果應用程式達到此臨界值，則需要佈建更多資源。您可以啟用自動調整規模或增加應用程式平行處理原則來達成此目 如需增加資源的詳細資訊，請參閱 <a href="#">在 Apache Flink 的受管理服務中實作應用程式調整</a> 。  |
| cpuUtilization > 臨界值        | 最大   | 這可以很好地指示應用程式的整體資源使用率，並且除非應用程式是 I/O 繫結，否則可用於主動擴充。正常情況下的最cpuUtilization 大預期大小，建議值為 80%。        | 您可以使用此指標來監視整個應用程式中工作管理員的最大使用CPU率。如果應用程式達到此臨界值，您就需要佈建更多資源您可以啟用自動調整規模或增加應用程式平行處理原則來達到此目的。如需增加資源的詳細資訊，請參閱 <a href="#">在 Apache Flink 的受管理服務中實作應用程式調整</a> 。 |
| threadsCount > 臨界值          | 最大   | 正常情況下的最threadsCount 大預期大小。   | 您可以使用此指標來監視整個應用程式中工作管理員中的執行緒洩漏。如果此量度達到此臨界值，請檢查您的應用程式程式碼是否有建立的執行緒而不關閉。   |

| 指標表達式  | 統計數字 | Threshold  | 描述   |
|--|------|--|--|
| <code>(oldGarbageCollectionTime * 100)/60_000 over 1 min period') &gt;</code> 臨界值      | 最大   | 最大預期oldGarbageCollectionTime 持續時間。我們建議您設定一個閾值，使典型的記憶體回收時間是指定臨界值的 60%，但應用程式的正確臨界值會有所不同。 | 如果此測量結果持續增加，這可能表示整個應用程式的工作管理員發生記憶體洩漏。                  |
| <code>RATE(oldGarbageCollectionCount) &gt;</code> 臨界值                                  | 最大   | 正常情況oldGarbageCollectionCount 下預期的最大值。您的應用程式的正確門檻會有所不同。                                | 如果此測量結果持續增加，這可能表示整個應用程式的工作管理員發生記憶體洩漏。                  |
| <code>Operator.currentOutputWatermark - Operator.currentInputWatermark &gt;</code> 臨界值 | 下限   | 正常情況下的最小預期浮水印增量。您的應用程式的正確門檻會有所不同。  | 如果此指標持續增加，這可能表示應用程式正在處理越來越舊的事件，或者上游子任務在越來越長的時間內未傳送浮水印。 |

## 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](#)
- [使用寫入 CloudWatch 記錄 SLF4J](#)

## 使用 Log4j 寫入 CloudWatch 日誌

1. 將下列相依性新增至應用程式的 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"
```

```
}
```

## 使用寫入 CloudWatch 記錄 SLF4J

1. 將下列相依性新增至應用程式的 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"
}
```



## 日誌管理服務的阿帕奇 Flink API 呼叫 AWS CloudTrail

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 會擷取 Apache Flink 受管理服務的所有 API 呼叫做為事件。擷取的呼叫包括來自 Apache Flink 主控台的受管理服務呼叫，以及針對 Apache Flink 作業對受管理服務的程式碼呼叫。API 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。您可以在帳戶中查看，搜索和下載最近的事 AWS 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 帳戶中持續的事件記錄 (包括 Apache Flink 的受管理服務的事件)，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，當您在主控台中建立追蹤時，追蹤會套用至所有 AWS 區域。追蹤記錄來自 AWS 分區中所有區域的事件，並將日誌檔傳送到您指定的 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 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。

每一筆事件或日誌專案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：

- 要求是使用 root 或 AWS Identity and Access Management (IAM) 使用者認證提出的。
- 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。

- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱[CloudTrail userIdentity](#)元素。

## 瞭解 Apache Flink 記錄檔項目的受管理服務

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": "kinesisanalytics.amazonaws.com",
      "eventName": "AddApplicationCloudWatchLoggingOption",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
      "requestParameters": {
        "applicationName": "cloudtrail-test",
        "currentApplicationVersionId": 1,
        "cloudWatchLoggingOption": {
          "logStreamARN": "arn:aws:logs:us-east-1:012345678910:log-group:cloudtrail-test:log-stream:flink-cloudwatch"
        }
      },
      "responseElements": {
        "cloudWatchLoggingOptionDescriptions": [
```

```

        {
            "cloudWatchLoggingOptionId": "2.1",
            "logStreamARN": "arn:aws:logs:us-east-1:012345678910:log-
group:cloudtrail-test:log-stream:flink-cloudwatch"
        }
    ],
    "applicationVersionId": 2,
    "applicationARN": "arn:aws: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": "kinesisanalytics.amazonaws.com",
    "eventName": "DescribeApplication",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
    "requestParameters": {
        "applicationName": "sample-app"
    },
    "responseElements": null,
    "requestID": "3e82dc3e-4470-11e9-9d01-e789c4e9a3ca",
    "eventID": "90ffe8e4-9e47-48c9-84e1-4f2d427d98a5",
    "eventType": "AwsApiCall",
    "apiVersion": "2018-05-23",
    "recipientAccountId": "012345678910"
}
]

```

```
}
```

# 調整阿帕奇 Flink 的 Amazon 託管服務中的性能

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

- [解決效能問題](#)
- [使用效能最佳做法](#)
- [監控效能](#)

## 解決效能問題

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。

## 瞭解資料路徑

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。默認的 Kryo 序列化程序可以處理任何可序列化類型，但是如果您的應用程序僅以類型存儲數據，則可以使用更高性能的序列化程序。POJO 如需有關 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](#)。

- 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

本節包含效能問題的潛在解決方案。

## 主題

- [CloudWatch 監視層級](#)

- [應用程式CPU量](#)
- [應用程式平行](#)
- [應用程式日誌記錄](#)
- [操作員並行](#)
- [应用逻辑](#)
- [應用記憶體](#)

## CloudWatch 監視層級

確認「CloudWatch 監視層級」未設定為太詳細的設定。

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[控制應用程式監控層](#)。

## 應用程式CPU量

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 auto 擴展，則在 15 分鐘內的使用量超過 75% 時，應CPU用程式會配置更多資源。如需擴展的詳細資訊，請參閱下面的[適當管理擴展](#)一節和[在 Apache Flink 的受管理服務中實作應用程式調整](#)。

### Note

應用程式只會根據CPU使用情況自動調整規模。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

增加應用程式的平行處理層級。您可以使用[UpdateApplication](#)動作的ParallelismConfigurationUpdate參數更新應用程式的平行處理原則。

根據預設，應用程式的最大值KPU為 64，並且可以透過要求提高限制來增加。

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[操作員並行](#)。

## 應用程式日誌記錄

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。如需如何讀取應用程式日誌的詳細資訊，請參閱[the section called “使用日誌見解分析 CloudWatch 日誌”](#)。

## 操作員並行

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](#)。

## 应用逻辑

檢查您的應用程式邏輯是否有效率低或性能不佳的操作，例如存取外部相依性 (例如資料庫或 Web 服務)，存取應用程式狀態等。如果外部相依性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性傳回 HTTP 500 錯誤。

如果您的應用程式使用外部相依性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/O](#)。

## 應用記憶體

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。

## 使用效能最佳做法

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。如果符合以下條件，您的應用程式KPU將自動增加：

- 已為應用程式啟用自動擴展。
- CPU在 15 分鐘內，使用率保持在 75% 以上。

如果已啟用自動調度資源，但CPU使用量未維持在此閾值，則應用程式將不會擴展。KPU如果您遇到不符合此臨界值的CPU使用量激增，或是不同使用量指標 (例如) 中出現尖峰的使用量heapMemoryUtilization，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[在 Apache Flink 的受管理服務中實作應用程式調整](#)。

### 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 [在 Apache Flink 的受管理服務中使用執行階段屬性](#) 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")
    );
```

如需子任務、任務空位和其他應用程式資源的相關資訊，請參閱[檢閱 Apache Flink 應用程式資源的受管理服務](#)。

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：

- [平行執行](#)
- [DataStream 转换](#)

## 監控外部相依性資源使用量

如果目的地發生效能瓶頸 (例如 Kinesis 串流、Firehose、DynamoDB 或 OpenSearch 服務)，您的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性。

### 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失敗，請檢查目的地服務的 CloudWatch 記錄檔是否有失敗。

## 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。

如需有關建立本機 Flink 安裝的資訊，請參閱 Apache Flink 說明文件中的[獨立版](#)。

## 監控效能

本節說明監控應用程式效能的工具。

### 使用指標監視 CloudWatch 效能

您可以使用指 CloudWatch 標來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 Apache Flink 應用程式的受管理服務搭配使用的相關資訊，請參閱[???](#)。

### 使用 CloudWatch 記錄和警示來監控效能

您可以使用 CloudWatch 記錄來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，例如資源使用或超過安全閾值的檢查點指標，或非預期的應用程式狀態變更。

如需為 Apache Flink 應用程式的受管理服務建立 CloudWatch 警示的相關資訊，請參閱[???](#)。

## 適用於 Apache Flink 和工作室筆記本配額的託管服務

### Note

阿帕奇 Flink 版本 1.6，1.8 和 1.11 一直沒有得到阿帕奇 Flink 社區的支持超過三年。我們計劃於 2024 年 11 月 5 日在 Amazon 管理服務中淘汰這些版本。從這個日期開始，您將無法為這些 Flink 版本建立新的應用程式。您目前可以繼續執行現有的應用程式。您可以使用適用於 Apache Flink 的 Amazon 受管服務中的就地版本升級功能，以狀態方式升級應用程式。如需詳細資訊，請參閱 [為 Apache Flink 使用就地版本升級](#)

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](#)。

- 根據預設，Kinesis 處理單元 (KPU) 的數量限制為 64 個。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。請確定您指定了需要套用新 KPU 限制的應用程式前置詞。

使用適用於 Apache Flink 的受管服務，您的 AWS 帳戶需支付已配置資源的費用，而不是應用程式使用的資源。我們會根據用於執行串流處理應用程式 KPU 的最大數量，按小時費率向您收費。單一記憶體可 KPU 為您提供 1 v CPU 和 4 GiB 的記憶體。對於每個 KPU，該服務還提供 50 GiB 的正在運行的應用程序存儲。

- 您可以為每個應用程式建立多達 1,000 個 Managed Service for Apache Flink [使用快照管理應用程式備](#)。
- 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

## 管理適用於 Apache Flink 的受管理服務的維護工作

Apache Flink 受管理服務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安全目標。AWS 下表列出 Managed Service for Apache Flink 執行此類維護的預設時段。應用程式維護可能會在與您的地區對應的時段內隨時發生。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相關資訊，請參閱[the section called “容錯能力：檢查點和儲存點”](#)。

若要變更 Apache Flink 受管理服務對您的應用程式執行維護的時間範圍，請使用 [UpdateApplicationMaintenanceConfigurationAPI](#)。

| 區域                  | 維護時段                   |
|---------------------|------------------------|
| AWS GovCloud (美國西部) | 上午 6 時至下午 2 時 UTC      |
| AWS GovCloud (美國東部) | 凌晨三時至十一時 UTC           |
| 美國東部 (維吉尼亞北部)       | 凌晨三時至十一時 UTC           |
| 美國東部 (俄亥俄)          | 凌晨三時至十一時 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)           | 下午一時至下午九時 UTC          |
| 亞太區域 (孟買)           | 下午 6 時 30 分至下午 3 時 UTC |
| 亞太區域 (海德拉巴)         | 下午 6 時 30 分至下午 3 時 UTC |
| 亞太區域 (首爾)           | 下午一時至下午九時 UTC          |
| 亞太區域 (新加坡)          | 下午 2 時至晚上 10 時 UTC     |
| 亞太區域 (悉尼)           | 12 時至晚上 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)           | 下午一時至下午九時 UTC          |

| 區域                    | 維護時段              |
|-----------------------|-------------------|
| 加拿大 (中部)              | 凌晨三時至十一時 UTC      |
| 中國 (北京)               | 下午一時至下午九時 UTC     |
| 中國 (寧夏)               | 下午一時至下午九時 UTC     |
| 歐洲 (法蘭克福)             | 上午 6 時至下午 2 時 UTC |
| 歐洲 (蘇黎世)              | 晚上 8 時至 4 時 UTC   |
| 歐洲 (愛爾蘭)              | 晚上 10 時至 6 時 UTC  |
| 歐洲 (倫敦)               | 晚上 10 時至 6 時 UTC  |
| 歐洲 (斯德哥爾摩)            | 上午十一時至七時 UTC      |
| 歐洲 (米蘭)               | 下午 9 時至 5 時 UTC   |
| 歐洲 (西班牙)              | 下午 9 時至 5 時 UTC   |
| 非洲 (開普敦)              | 晚上 8 時至 4 時 UTC   |
| 歐洲 (愛爾蘭)              | 晚上 10 時至 6 時 UTC  |
| 歐洲 (倫敦)               | 上午十一時至七時 UTC      |
| Europe (Paris)        | 上午十一時至七時 UTC      |
| 歐洲 (斯德哥爾摩)            | 上午十一時至七時 UTC      |
| Middle East (Bahrain) | 下午一時至下午九時 UTC     |
| 中東 (UAE)              | 下午 6 時至 2 時 UTC   |
| 南美洲 (聖保羅)             | 下午 7 時至 3 時 UTC   |
| 以色列 (特拉維夫)            | 晚上 8 時至 4 時 UTC   |

## UUID為所有運營商設置一個

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。其中一個問題是運算子 ID 不符。Flink 預期 Flink 工作圖表運算子具有明確、一致 IDs 的運算子。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 IDs 來唯一識別工作圖中的運算子，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與儲存點中 IDs 定義的運算子 IDs 之間找不到 1:1 對應時，就會發生運算子 ID 不符問題。當未設定明確一致運算子 IDs 且 Flink 自動產生可能與每個工作圖形建立不一致 IDs 的運算子時，就會發生這種情況。在維護執行期間，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 代碼中 UUID 為所有運營商設置。如需詳細資訊，請參閱[生產準備就緒下 UUID 的所有操作員設定 a 主題](#)。

## 識別應用程式何時進行維護

您可以使用來查看 Apache Flink 的受管理服務是否已對您的應用程式執行維護動作。ListApplicationOperations API

以下是可協助您篩選清單 ListApplicationOperations 以進行應用程式維護的範例要求：

```
{
  "ApplicationName": "MyApplication",
  "operation": "ApplicationMaintenance"
}
```

# 為您的 Apache Flink 應用程式的受管理服務實現生產準備

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。

## 負載測試您的應用程式

應用程式的某些問題僅在高負載下才會表現出來。我們已經看到了應用程序似乎健康的情況，但一個操作事件顯著放大了應用程序的負載。這可能完全獨立於應用程序本身。如果資料來源或資料接收器在幾個小時內無法使用，則 Flink 應用程式將無法進行。修復該問題後，積存積壓的未處理數據已積累，這可能會完全耗盡可用的可用資源。然後，負載可以放大以前沒有出現的錯誤或性能問題。

因此，您必須針對生產應用程式執行適當的負載測試。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。然後啟動應用程序並讓它從一開始就消耗數據。例如，如果是 Kinesis 資料串流，請使用的起始位置。TRIM\_HORIZON

## 定義最大平行度

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。

最大平行處理原則是在第一次建立狀態時設定的。

依預設，最大平行原則設定為：

- 128：如果平行處理層級  $\leq 128$
- $\text{MIN}(\text{nextPowerOfTwo}(\text{parallelism} + (\text{parallelism} / 2)), 2^{15})$ ：如果平行處理層級  $> 128$



如果您打算將應用程式擴展到 128 個以上的平行處理原則，您應該明確定義最大平行處理原則。

您可以使用 `env.setMaxParallelism(x)` 或單一運算子在應用程式層級定義最大平行處理原則。除非指定不同，否則所有運算子都會繼承應用程式的最大平行原則。

如需詳細資訊，請參閱 Apache Flink 文件中的 [設定最大平行處理原則](#)。

## UUID 為所有運營商設置一個

在 Flink 將儲存點對映回個別運算子的作業中會使用 A。UUID 為每個運算子設定一個特定的對應可提供要還原的儲存點程序的穩定對應。

```
.map(...).uid("my-map-function")
```

如需詳細資訊，請參閱 [生產就緒性檢查清單](#)。

# 為 Apache Flink 應用程式維護受管理服務的最佳做法

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](#)
- [不受支援的連接器版本](#)
- [效能與平行處理層級](#)
- [設定每個運算子的平行處理層級](#)
- [日誌](#)
- [編碼](#)
- [管理憑證](#)
- [從具有較少碎片/分割區的來源讀取](#)
- [Studio 筆記本重新整理間隔](#)
- [Studio 筆記本最佳效能](#)
- [浮水印策略和閒置碎片如何影響時間範圍](#)
- [UUID為所有運營商設置一個](#)
- [添加 ServiceResourceTransformer 到 Maven 的陰影插件](#)

## 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。如需排程維護的相關資訊，請參閱[管理適用於 Apache Flink 的受管理服務的維護工作](#)。
- 設置 `ApplicationSnapshotConfiguration`：`enableSnapshotsEnabled` 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 `enableSnapshotsEnabled` 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相關資訊，請參閱 [???。](#)

如需實作容錯能力的詳細資訊，請參閱 [在 Apache Flink 的受管理服務中實作容錯能力。](#)

## 不受支援的連接器版本

從 Apache Flink 1.15 版或更新版本起，如果應用程式使用捆綁在應用程式中的不支援 Kinesis 連接器版本，則 Apache Flink 的受管理服務會自動防止應用程式啟動或更新。JARs 升級至 Apache Flink 版本 1.15 或更新版本的受管理服務時，請確定您使用的是最新的 Kinesis 連接器。這是指 1.15.2 版本或更新版本。Apache Flink 的受管理服務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使用儲存點停止功能失敗，進而防止正常停止/更新作業。若要進一步了解適用於 Apache Flink 版本的 Amazon 受管服務中的連接器相容性，請參閱 [Apache Flink 連接器](#)。

## 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](#) 用代碼更改每個運算符的並行性。
- 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。如需設定運算子平行處理的詳細資訊，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 [在 Apache Flink 的受管理服務中實作應用程式調整](#)。

## 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 `using` 覆寫單一運算子的平行處理原則。DataStream API `.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.setParallelism(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。如需詳細資訊，請參閱 [???](#)

如需實作記錄的詳細資訊，請參閱 [???](#)。

## 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。
- 考慮著色您的應用程序JAR文件及其包含的依賴項。如果應用程式與 Apache Flink 執行期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。有關為應用程序JAR文件著色的更多信息，請參閱 [Apache Maven 陰影插件](#)。

## 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。如果未與原生整合的服務需要驗證IAM，例如需要驗證使用者名稱和密碼的資料庫，您應該考慮將密碼儲存在 [AWS Secrets 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。

某些運算子會使用多個輸入串流；例如，聯集或是跟隨 `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(1 -> 01)
    .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.splititerator(), false).count();
            long timestamp = context.currentWatermark();

            System.out.print("XXXXXXXXXXXXXXXXX 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:
49627894338503151834508157912666084957565273949901684850,}}'}, 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:
49627894338570054070103749782090692112383219034419626146,}}'}, 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:
49627894338503151834508157912666084957565273949901684850,}}'} 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:
49627894338458550344111096666383013521019977226889723986,}}'}, 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:
49627894338458550344111096666383013521019977226889723986,}}'} 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:
85070591730234615865843651857942052864,EndingHashKey:
106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
49627894338436249598912566043241477802747328865383743554,}}'}, 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:
63802943797675961899382738893456539648,EndingHashKey:
85070591730234615865843651857942052863},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:
319014718988379809496913694467282698240,EndingHashKey:
340282366920938463463374607431768211455},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:
297747071055821155530452781502797185024,EndingHashKey:
319014718988379809496913694467282698239},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:
106338239662793269832304564822427566079},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:
191408831393027885698148216680369618944,EndingHashKey:
212676479325586539664609129644855132159},SequenceNumberRange: {StartingSequenceNumber:
49627894338547753324905219158949156394110570672913645714,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
21267647932558653966460912964485513215},SequenceNumberRange: {StartingSequenceNumber:
49627894338347046618118443550675334929656735419359821826,}}'}, 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:
49627894338614655560500811028373763548928515757431587010,}}'}, 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:
49627894338525452579706688535807620675837922311407665282,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey:
21267647932558653966460912964485513216,EndingHashKey:
42535295865117307932921825928971026431},SequenceNumberRange: {StartingSequenceNumber:
49627894338369347363316974173816870647929383780865802258,}}'} 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:
49627894338592354815302280405232227830655867395925606578,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
```

```
Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
21267647932558653966460912964485513215},SequenceNumberRange: {StartingSequenceNumber:
49627894338347046618118443550675334929656735419359821826,}}'} 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:
276479423123262501563991868538311671808,EndingHashKey:
297747071055821155530452781502797185023},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:
191408831393027885698148216680369618944,EndingHashKey:
212676479325586539664609129644855132159},SequenceNumberRange: {StartingSequenceNumber:
49627894338547753324905219158949156394110570672913645714,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,244 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 6 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000010,HashKeyRange: {StartingHashKey:
212676479325586539664609129644855132160,EndingHashKey:
233944127258145193631070042609340645375},SequenceNumberRange: {StartingSequenceNumber:
49627894338570054070103749782090692112383219034419626146,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 1
event: 6; timestamp: 1648030822428, 2022-03-23T10:20:22.428Z
```

```
11:59:23,377 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 3 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000015,HashKeyRange: {StartingHashKey:
319014718988379809496913694467282698240,EndingHashKey:
340282366920938463463374607431768211455},SequenceNumberRange: {StartingSequenceNumber:
49627894338681557796096402897798370703746460841949528306,}}}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,405 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 2 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000014,HashKeyRange: {StartingHashKey:
297747071055821155530452781502797185024,EndingHashKey:
319014718988379809496913694467282698239},SequenceNumberRange: {StartingSequenceNumber:
49627894338659257050897872274656834985473812480443547874,}}}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,581 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000008,HashKeyRange: {StartingHashKey:
170141183460469231731687303715884105728,EndingHashKey:
191408831393027885698148216680369618943},SequenceNumberRange: {StartingSequenceNumber:
49627894338525452579706688535807620675837922311407665282,}}}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,586 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 1 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000013,HashKeyRange: {StartingHashKey:
276479423123262501563991868538311671808,EndingHashKey:
297747071055821155530452781502797185023},SequenceNumberRange: {StartingSequenceNumber:
49627894338636956305699341651515299267201164118937567442,}}}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:24,790 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000012,HashKeyRange: {StartingHashKey:
255211775190703847597530955573826158592,EndingHashKey:
276479423123262501563991868538311671807},SequenceNumberRange: {StartingSequenceNumber:
49627894338614655560500811028373763548928515757431587010,}}}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 2
event: 4; timestamp: 1648030809282, 2022-03-23T10:20:09.282Z
event: 3; timestamp: 1648030797697, 2022-03-23T10:19:57.697Z
event: 5; timestamp: 1648030810871, 2022-03-23T10:20:10.871Z
```

```

11:59:24,907 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000011,HashKeyRange: {StartingHashKey:
233944127258145193631070042609340645376,EndingHashKey:
255211775190703847597530955573826158591},SequenceNumberRange: {StartingSequenceNumber:
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
XXXXXXXXXXXXXXXXX Window with 3 events; Watermark: 1648030809281, 2022-03-23T10:20:09.281Z
3
1
2
XXXXXXXXXXXXXXXXX Window with 2 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
4
5
XXXXXXXXXXXXXXXXX Window with 1 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
6
XXXXXXXXXXXXXXXXX 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))`)。



## UUID為所有運營商設置一個

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。其中一個問題是運算子 ID 不符。Flink 預期 Flink 工作圖表運算子具有明確、一致 IDs 的運算子。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 IDs 來唯一識別工作圖中的運算子，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與儲存點中 IDs 定義的運算子 IDs 之間找不到 1:1 對應時，就會發生運算子 ID 不符問題。當未設定明確一致運算子 IDs 且 Flink 自動產生可能與每個工作圖形建立不一致 IDs 的運算子時，就會發生這種情況。在維護執行期間，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 代碼中 UUID 為所有運營商設置。如需詳細資訊，請參閱 [\[生產就緒\] 下的 \[UUID 為所有操作員設定 a\]](#) 主題。

## 添加 ServiceResourceTransformer 到 Maven 的陰影插件

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。使用多個 Flink 相依性 SPI 可能會導致 [uber-jar 發生衝突](#)，以及 [非預期的應用程式行為](#)。建議添加 Maven [ServiceResourceTransformer](#) 的陰影插件，在 pom.xml 中定義

```
<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>
  </plugins>
</build>
```

```
        </executions>  
    </plugin>
```

## 阿帕奇 Flink 有狀態函數

[有狀態函數](#)是一種可簡化建置分散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，介於 Kubernetes 叢集和 Managed Service for Apache Flink 的封裝有狀態函數之間有幾項差異。有狀態函數應用程式的最重要方面是[模組組態](#)包含設定有狀態函數執行期所需的所有執行期資訊。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 for Apache Flink 無法如此。

以下是 StateFun Python 的例子為阿帕奇 Flink 管理服務的適應：

## 阿帕奇 Flink 應用程序模板

客戶可以編譯只調用有狀態函數執行期並且包含必要相依性的 Flink 應用程式 jar，而不是使用客戶容器作為有狀態函數執行期。對於 Flink 1.13，必要相依性如下所示：

```
<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期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

有狀態函數模組組態需要包含在類別路徑中，才能讓有狀態函數執行期發現。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。

# 阿帕奇 Flink 設置

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 Apache Flink 應用程式的受管理服務在程式碼中設定，而其他值則無法變更。

您可以使用本節中的連結進一步瞭解 Apache flink 設定，以及哪些設定可以修改。

本主題包含下列章節：

- [阿帕奇 Flink 配置](#)
- [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 內容](#)

## 阿帕奇 Flink 配置

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，若要調整 Flink 組態屬性，以改善某些具有高平行度、高記憶體和狀態使用率的應用程式的效能，或在 Apache Flink 中啟用新的除錯功能，您可以透過要求支援案例來變更某些特性。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。阿帕奇 Flink 的託管服務使用 RocksDBState 後端。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

如需狀態後端的詳細資訊，請參閱 Apache Flink 文件中的[狀態後端](#)。

## 檢查點

Managed Service for Apache Flink 使用具有下列值的預設檢查點組態。其中一些值可以使用變更[CheckpointConfiguration](#)。您必須針對 Apache Flink CUSTOM 的受管理服務設定 `CheckpointConfiguration.ConfigurationType` 為 `CUSTOM`，才能使用修改的檢查點值。

| 設定                                      | 可以修改嗎？ | 方法   | 預設值   |
|---|--------|--|-------|
| <code>CheckpointingEnabled</code>       | 可修改    | <a href="#">建立應用程式</a><br><a href="#">更新應用程式</a><br><a href="#">AWS CloudFormation</a> | True  |
| <code>CheckpointInterval</code>         | 可修改    | <a href="#">建立應用程式</a><br><a href="#">更新應用程式</a><br><a href="#">AWS CloudFormation</a> | 60000 |
| <code>MinPauseBetweenCheckpoints</code> | 可修改    | <a href="#">建立應用程式</a><br><a href="#">更新應用程式</a><br><a href="#">AWS CloudFormation</a> | 5000  |
| 未對齊的檢查點                                 | 可修改    | <a href="#">支援案例</a>   | False |
| 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 |

| 設定        | 可以修改嗎？ | 方法  | 預設值                              |
|-----------|--------|-----|----------------------------------|
| 保留的最大檢查點  | 不可修改   | N/A | 1                                |
| 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。 |

## 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。您可以將應用程式的 `AllowNonRestoredState` 參數設定為 `true`，以允許作業成功。 [FlinkRunConfiguration](#) 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 [Apache Flink 文件](#) 中的 [允許非還原的狀態](#)。

## 堆積大小

適用於 Apache Flink 的受管理服務會分配每 KPU 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 “在 Apache Flink 的受管理服務中實作應用程式調整”](#)。

如需有關 JVM 堆積大小的詳細資訊，請參閱 [Apache Flink 文件](#) 中的 [組態](#)。

## 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](#) 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。如果您要修改此清單以外還有其他 Flink 組態屬性，請在您的情況下指定確切的屬性。

## 重啟策略

從 Flink 1.19 及更高版本開始，我們默認使用exponential-delay重新啟動策略。依預設，所有先前版本都使用fixed-delay重新啟動策略。

restart-strategy:

restart-strategy.fixed-delay.delay:

restart-strategy.exponential-delay.backoff-multiplier:

restart-strategy.exponential-delay.initial-backoff:

restart-strategy.exponential-delay.jitter-factor:

restart-strategy.exponential-delay.reset-backoff-threshold:

## 檢查點和狀態後端

state.backend:

state.backend.fs.memory-threshold:

state.backend.incremental:

## 檢查點

execution.checkpointing.unaligned:

execution.checkpointing.interval-during-backlog:

## RockSDB 原生度量

RockSDB 原生測量結果不會運送至。CloudWatch啟用後，您可以從 Flink 儀表板或RESTAPI使用自訂工具的 Flink 存取這些指標。

適用於 Apache Flink 的受管理服務可讓客戶使用以唯讀模式存取最新的 Flink [RESTAPI](#)(或您使用的支援版本)。[CreateApplicationPresignedUrlAPI](#)Flink 自己的儀表板會使用此API功能，但也可以由自訂監控工具使用。

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/阿卡

`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

`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:`

## 端點和用戶REST端的進階選項

`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 網頁使用者介面的進階選項

`web.timeout:`

## 檢視已設定的 Flink 內容

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。

# 為 Apache Flink 設定受管服務，以存取 Amazon 中的資源 VPC

您可以將 Apache Flink 應用程式的受管理服務設定為連線到帳戶中虛擬私有雲 (VPC) 中的私有子網路。使用 Amazon Virtual Private Cloud (AmazonVPC) 為資源 (例如資料庫、快取執行個體或內部服務) 建立私有網路。將您的應用程式 Connect VPC 到，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 應用程式權](#)
- [適用於 Apache Flink 應用程VPC式的已連線受管理服務的網際網路和服務存取](#)
- [使用適用於阿帕奇 Flink 的受管理服務 VPC API](#)
- [範例：使用VPC來存取 Amazon MSK 叢集中的資料](#)

## Amazon VPC 概念

Amazon VPC 是 Amazon 的網路層 EC2。如果您是 Amazon 新手 EC2，請參閱 [Amazon 是什麼 EC2?](#) 請參閱 Amazon Linux 執行個體 EC2 使用者指南，以取得簡短的概觀。

以下是關鍵概念 VPCs：

- 虛擬私有雲 (VPC) 是專用於您 AWS 帳戶的虛擬網路。
- 子網路是您的 IP 位址範圍 VPC。
- 「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。
- 網際網路閘道是水平擴充、備援且高可用性的 VPC 元件，可讓您與網際網路中的 VPC 執行個體之間進行通訊。因此不會對網路流量強加可用性風險或頻寬限制。
- VPC 端點可讓您以私密方式連線 VPC 至受支援的 AWS 服務和 VPC 端點服務，PrivateLink 而不需要網際網路閘道、NAT 裝置、VPN 連線或 AWS Direct Connect 連線。您中的執行個體 VPC 不需要公用 IP 位址即可與服務中的資源通訊。您 VPC 與其他服務之間的流量不會離開 Amazon 網路。

如需 Amazon VPC 服務的詳細資訊，請參閱 [Amazon Virtual Private Cloud 使用者指南](#)。

Apache Flink 的受管理服務會在您為應用程式 VPC 設定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，具體取決於每個 KPU 應用程式的平行處

理原則和平行處理原則。如需應用程式擴展的詳細資訊，請參閱[在 Apache Flink 的受管理服務中實作應用程式調整](#)。

#### Note

VPC應用程式不支SQL援組態。

#### Note

Apache Flink 服務的受管理服務會管理具有VPC組態之應用程式的檢查點和快照集狀態。

## VPC應用程式權

本節說明您的應用程式將需要使用的權限原則VPC。如需如何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](#)。

下列權限原則會授與您的應用程式與VPC。若要使用此許可政策，請將其新增至應用程式的執行角色。

### 添加訪問 Amazon 的許可政策 VPC

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VPCReadOnlyPermissions",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeDhcpOptions"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ENIReadWritePermissions",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
```

```
    "ec2:CreateNetworkInterfacePermission",
    "ec2:DescribeNetworkInterfaces",
    "ec2:DeleteNetworkInterface"
  ],
  "Resource": "*"
}

]
}
```

### Note

使用主控台 (例如 CloudWatch 日誌或 AmazonVPC) 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。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。

## 適用於 Apache Flink 應用程VPC式的已連線受管理服務的網際網路和服務存取

根據預設，當您將 Apache Flink 應用程式的受管理服務連線到您帳戶VPC中的一個時，除非VPC提供存取權，否則它無法存取網際網路。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必須包含公用子網路中的NAT閘道或執行個體。
- 從私人子網路到公用子網路中NAT閘道的輸出流量必須存在路由。

### Note

有幾個服務提供[VPC端點](#)。您可以使用VPC端點從內部連接到 Amazon 服務，VPC而無需存取網際網路。

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-...)。

使用公用子網路 (VPC使用 aNAT) 和一或多個私有子網路進行設定之後，請執行下列動作以識別您的私人和公用子網路：

- 在VPC主控台中，從導覽窗格中選擇「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：

- 在 <https://console.aws.amazon.com/flink> 開啟適用於阿帕奇 Flink 的受管理服務
- 在 Managed Service for Apache Flink 頁面，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，選擇設定。
- 在「VPC連線」區段中，選擇VPC要與您的應用程式建立關聯的。選擇與您要應用程式用來存取VPC資源VPC的子網路和安全性群組相關聯。
- 選擇更新。

## 相關資訊

[VPC使用公用和私有子網路建立](#)

[NAT閘道基礎](#)

## 使用適用於阿帕奇 Flink 的受管理服務 VPC API

您可以使用下列 Apache Flink API 作業的受管理服務來管理您的應VPCs用程式。如需有關使用 Apache Flink 的受管理服務的資訊API，請參閱[API 範例程式碼](#)。

## 建立應用程式

使用此[CreateApplication](#)動作可在建立期間將VPC組態新增至應用程式。

下列CreateApplication動作的要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

使用此[AddApplicationVpcConfiguration](#)動作可在建立應VPC用程式之後將其新增至應用程式。

下列AddApplicationVpcConfiguration動作的要求程式碼範例會將VPC設定新增至現有應用程式：

```

{
  "ApplicationName": "MyApplication",

```

```
"CurrentApplicationVersionId": 9,
"VpcConfiguration": {
  "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
  "SubnetIds": [ "subnet-0123456789abcdef0" ]
}
}
```

## DeleteApplicationVpcConfiguration

使用此[DeleteApplicationVpcConfiguration](#)動作可從應用程式中移除VPC組態。

下列AddApplicationVpcConfiguration動作的範例要求程式碼會從應用程式移除現有的VPC組態：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 9,
  "VpcConfigurationId": "1.1"
}
```

## 更新應用

使用此[UpdateApplication](#)動作可一次更新所有應用程式的VPC組態。

下列UpdateApplication動作的要求程式碼範例會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來存取 Amazon MSK 叢集中的資料

如需有關如何在叢集中存取 Amazon MSK 叢集資料的完整教學課程VPC，請參閱[MSK 複寫](#)。

# 疑難排解 Apache Flink 的受管理服務

下列主題可協助您疑難排解使用適用於 Apache Flink 的 Amazon 受管服務時可能遇到的問題。

選擇適當的主題來檢閱解決方案。

## 主題

- [開發疑難排](#)
- [運行時疑難](#)

## 開發疑難排

本節包含診斷及修正 Apache Flink 應用程式受管理服務之開發問題的相關資訊。

## 主題

- [系統復原最佳做法](#)
- [胡迪配置最佳實踐](#)
- [Apache Flink 火焰圖](#)
- [EFO連接器 1.15.2 的認證提供者問題](#)
- [應用程式使用不支援的 Kinesis 連接器](#)
- [編譯錯誤：「無法解析項目的依賴關係」](#)
- [無效的選擇：「動力分析 Sv2」](#)
- [UpdateApplication 動作不重新加載應用程序代碼](#)
- [S3 StreamingFileSink FileNotFoundExceptions](#)
- [FlinkKafkaConsumer 與保存點停止問題](#)
- [FLINK 1.15 非同步接收器死鎖](#)
- [在重新分片期間，Amazon Kinesis 資料串流來源處理不順序](#)

## 系統復原最佳做法

透過適用於 Apache Flink 的 Amazon 受管服務中的自動系統復原和操作可見性功能，您可以識別並解決應用程式的問題。

## 系統復原

如果您的應用程式更新或擴展操作因為客戶錯誤 (例如程式碼錯誤或權限問題) 而失敗，如果您選擇使用此功能，則 Apache Flink 的 Amazon 受管服務會自動嘗試回復到之前執行的版本。如需詳細資訊，請參閱[為您的 Apache Flink 應用程式的受管理服務啟用系統復原](#)。如果此自動返回失敗，或者您尚未選擇加入或選擇退出，您的應用程序將進入狀態。READY若要更新您的應用程式，請完成以下步驟：

### 手動復原

如果應用程式未進行中且處於暫時狀態很長時間，或者應用程式已成功轉移至Running，但您看到下游問題 (例如已成功更新 Flink 應用程式中的處理錯誤)，您可以使用 RollbackApplication API

1. 呼叫 RollbackApplication-這將恢復到以前的運行版本並恢復以前的狀態。
2. 使用監視復原作業DescribeApplicationOperationAPI。
3. 如果還原失敗，請使用先前的系統復原步驟。

### 營運可見性

ListApplicationOperationsAPI會顯示應用程式上所有客戶和系統作業的歷史記錄。

1. 從清單中operationId取得失敗作業的。
2. 呼叫DescribeApplicationOperation並檢查狀態和 statusDescription。
3. 如果作業失敗，描述會指向要調查的潛在錯誤。

常見錯誤代碼錯誤：使用回滾功能恢復到上一個工作版本。解決錯誤並重試更新。

權限問題：使用DescribeApplicationOperation查看所需的權限。更新應用程式權限並重試。

針對 Apache Flink 服務問題的 Amazon 受管服務：檢查 AWS Health Dashboard 或開啟支援案例。

## 胡迪配置最佳實踐

若要在 Apache Flink 的受管理服務上執行 Hudi 連接器，我們建議您進行下列組態變更。

停用 hoodie.embed.timeline.server

Flink 上的 Hudi 連接器在 Flink 工作管理器 (JM) 上設置嵌入式時間軸 (TM) 服務器以緩存元數據，以在工作平行性高時提高性能。建議您在 Apache Flink 的受管理服務上停用此內嵌式伺服器，因為我們會停用 JM 與 TM 之間的非 FLink 通訊。

如果啟用此伺服器，Hudi 寫入會先嘗試連線到 JM 上的嵌入式伺服器，然後再從 Amazon S3 讀取中繼資料。這表示 Hudi 會導致連線逾時，延遲 Hudi 寫入，並對 Apache Flink 的受管理服務造成效能影響。

## 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 文件](#) 中所述。

如果要為您的應用程式禁用 Flame 圖，請創建一個案例以請求將其禁用為您的應用程式 ARN。如需詳細資訊，請參閱 [AWS 支援中心](#)。

## EFO 連接器 1.15.2 的認證提供者問題

Kinesis Data Streams EFO 連接器版本最高為 1.15.2 存在一個 [已知問題](#)，其中 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。這可能會導致使用跨帳戶存取 Kinesis 使 EFO 用連接器發生問題。

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

適用於 Apache Flink 1.15 版或更新版本的 Apache Flink 受管理服務，如果應用程式使用捆綁到 [應用程式或歸檔中的不支援 Kinesis 連接器版本 \(1.15.2 之前版本\)](#)，則會自動拒絕應用程式啟動或更新。JARs ZIP

### 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 相依性。如果使用 Maven 作為專案的建置工具，請按照 [更新 Maven 相依性](#) 操作。如果使用 Gradle，請按照 [更新 Gradle 相依性](#) 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，請建立一個支援票證。

## 更新 Maven 相依性

1. 開啟專案的 pom.xml。
2. 尋找專案的相依性。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 相依性

1. 開啟專案的 `build.gradle` (或針對 Kotlin 應用程式的 `build.gradle.kts`)。
2. 尋找專案的相依性。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：「無法解析項目的依賴關係」

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，您必須下載連接器的 Apache Flink 原始程式碼 (版本 1.8.2 來源 <https://flink.apache.org/downloads.html>)。如需如何下載、編譯和安裝 Apache Flink 來源程式碼的相關指示，請參閱 [the section called “使用阿帕奇 Flink Kinesis 串流連接器與先前的 Apache Flink 版本”](#)。

## 無效的選擇：「動力分析 Sv2」

若要使用適用於 Apache Flink 的受管理服務的 v2API，您需要最新版本的 AWS Command Line Interface (AWS CLI)。

若要取得有關升級的資訊 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》AWS Command Line Interface 中的 [〈安裝〉](#)。

## UpdateApplication 動作不重新加載應用程式代碼

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` 會將記錄寫入檔案系統支援的[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，並在寫入資料時新增檔案。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。

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 — 只有停止或更新應用程式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](#) 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，就可能發生這種情況。為避免此類失敗，請確保 S3 儲存貯體 MPU 生命週期政策涵蓋的使用案例足夠長的期間。
  - 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 Apache Flink 版本 1.15 的受管理服務中修正。將您的應用程式遷移至最新版本的 Apache Flink 受管理服務，以防止重複發生。我們也建議 `StreamingFileSink` 您從移轉至 [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 `FlinkKafkaConsumer`，您的應用程序可能會卡在中 `UPDATING`，`STOPPING` 或者 `SCALING`，如果您啟用了系統快照。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。

如果您正在使用啟 `FlinkKafkaConsumer` 用快照的情況下，Flink 工作可能會使用保存點 API 請求處理停止，則 `FlinkKafkaConsumer` 可能會失敗並報告執行時期錯誤。 `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`
- 連結/SQL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。

該問題是由一個錯誤引起的，AWS SDK導致在使用異步HTTP客戶端時不會向調用者顯示某些錯誤。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 2.20.144 版 AWS SDK開始，已修正此問題。

以下是如何更新受影響的連接器，以便在應用程式 AWS SDK中使用新版本的指示：

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 連接器使用陰影將某些相依性 (包括) 封裝到連接器 jar 中。AWS SDK若要更新 AWS SDK版本，請使用下列程序來取代這些著色類別：

## Maven

1. 將 Kinesis 連接器和所需 AWS SDK模組新增為專案相依性。
2. 設定 maven-shade-plugin：
  - a. 新增篩選器以在複製 Kinesis 連接器 jar 的內容時排除著色 AWS SDK類別。
  - b. 新增重新定位規則以將更新的 AWS SDK類別移至封裝，這是 Kinesis 連接器所預期的。

### 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 AWS SDK 模組新增為專案相依性。
2. 調整 shadowJar 配置：
  - a. 複製 Kinesis 連接器罐的內容時，請排除著色 AWS SDK 類別。
  - b. 將更新的 AWS SDK 類別重新定位至 Kinesis 連接器所預期的套件。

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("software/amazon/kinesis/shaded/software/amazon/awssdk/**/*")
    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，應該在項目構建配置中強制執行該 SDK 版本。

## Maven

將材料 AWS SDK 表 (BOM) 加入至檔案的相依性管理區段，以強制執行專 pom.xml 案的 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

在材料 AWS SDK 表 (BOM) 上加入平台相依性，以強制執行專案的 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 相依性作封裝為單個 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，指定所需的 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，指定所需的 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，指定所需的 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，指定所需的 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：

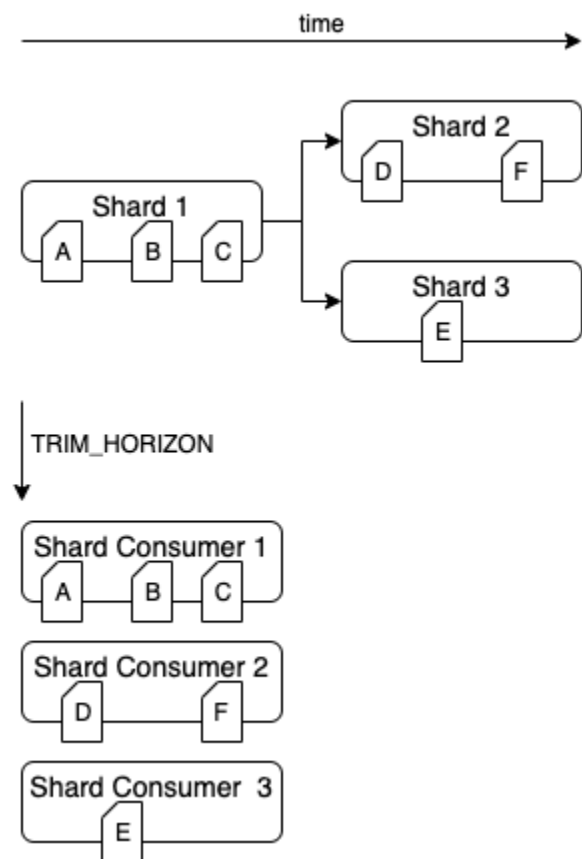
```
mvn clean install -DskipTests -Dfast -Dflink.version=1.15.4 -  
Daws.sdk.version=2.20.144
```

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 資料串流來源處理不順序

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 應用程式可以容忍處理延遲，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。

## 運行時疑難

本節包含診斷和修正 Managed Service for Apache Flink 應用程式執行期問題的相關資訊。

主題

- [故障診斷工具](#)
- [應用問題](#)
- [應用程式重新啟](#)
- [輸送量太慢](#)
- [无界状态增长](#)
- [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](#)

## 故障診斷工具

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 測量結果的臨界值，以指出應用程式中的錯誤或瓶頸狀況。如需建議 CloudWatch 警示的相關資訊，請參閱[搭配 Amazon 管理服務搭配 Apache Flink 使用 CloudWatch 警示](#)。

## 應用問題

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](#)
- [無法訪問中的資源 VPC](#)
- [寫入 Amazon S3 儲存貯體時資料遺失](#)
- [應用程式處於RUNNING狀態，但未處理資料](#)
- [快照、應用程式更新或應用程式停止錯誤：InvalidApplicationConfigurationException](#)
- [Java 的 .nio. 文件。NoSuchFileException: /usr /本地/開放jdk-8/lib/安全/緩存器](#)

## 應用程式卡在暫時狀態

如果您的應用程式維持暫時狀態 (STARTING/UPDATING/STOPPING、或AUTOSCALING)，您可以使用Force參數設定為的[StopApplication](#)動作來停止應用程式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。如需有關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遺失或重複。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：請確認您的應用程式JAR檔案小於 512 MB。JAR不支援大於 512 MB 的檔案。
- 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。若要變更此行為並允許還原成功，請將應用程式的AllowNonRestoredState參數變更[FlinkRunConfiguration](#)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)。如果您的工作無法在此逾時內啟動，您將會看到如下 CloudWatch 記錄：

```
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 日誌以及您的應用程式代碼，以查看該 main 方法是否會出現此延遲。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[使用快照管理應用程式備](#)。
- 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。如需調整運算子效能的相關資訊，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`。

## 無法訪問中的資源 VPC

如果您的應用程式使用在 Amazon 上 VPC 執行 VPC，請執行下列動作以確認您的應用程式是否可以存取其資源：

- 檢查您的 CloudWatch 日誌是否有以下錯誤。此錯誤表示您的應用程式無法訪問以下內容中的資源 VPC：

```
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。

如需有關設定和分析 CloudWatch 記錄檔的資訊，請參閱[在 Amazon 管理服務中為 Apache Flink 進行日誌記錄和監控](#)。

## 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。要使用阿帕奇 Flink 1.6.2 寫入 Amazon S3 存儲桶，我們建議使用 Firehose。如需有關搭配 Apache Flink Firehose 理服務搭配使用的詳細資訊，請參閱[Firehose 水槽](#)

## 應用程式處於RUNNING狀態，但未處理資料

您可以使用 [ListApplications](#) 或 [DescribeApplication](#) 動作來檢查應用程式狀態。如果您的應用程式進入RUNNING狀態，但未將資料寫入接收器，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 Apache Flink 的受管理服務主控台中執行此操作，也可以使用動作的 `SnapshotsEnabledUpdate` 參數來執行此 [UpdateApplication](#) 操作。
- 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 的 .nio. 文件。NoSuchFileException: /usr /本地/開放jdk-8/lib/安全/緩存器

SSL信任庫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

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`)。您可以使用下列日誌深入解析查詢，查詢應用程式 CloudWatch 記錄檔是否有這些狀態變更：[分析錯誤：應用程式工作相關失敗](#)。

### 原因和解決方案

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：

- 運算符拋出異常：如果未處理應用程序中的操作符中的任何異常，則應用程序將失敗 ( 通過解釋操作符無法處理失敗 )。應用程式會從最新的檢查點重新啟動，以維護「恰好一次」的處理語義。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：應用程式工作相關失敗”](#)。

- Kinesis 資料串流未正確佈建：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](#)是 `ReadProvisionedThroughputExceeded` 否 `WriteProvisionedThroughputExceeded` 有錯誤。

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、未遇到讀取或寫入節流，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，則該運算子可能會超載，而應用程式可能會損毀。如需調整運算子平行處理的相關資訊，請參閱[適當管理運算子擴展](#)。
- 應用程式失敗 `DaemonException`：如果您使用的是 1.11 之前的 Apache Flink 版本，則應用程式記錄檔中會顯示此錯誤。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 `TimeoutException`，顯示 `FlinkException`、或 `RemoteTransportException`：如果工作管理員當機，這些錯誤可能會出現在應用程式記錄檔中。如果應用程式超載，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：

- 檢查您的 CloudWatch 指標是否有 CPU 或記憶體使用量的異常峰值。
- 檢查應用程式以了解是否有輸送量問題。如需詳細資訊，請參閱[解決效能問題](#)。
- 檢查應用程式日誌以了解是否有應用程式程式碼所引發的未處理例外狀況。
- 應用程式失敗並顯示「找 `JaxbAnnotationModule` 不到」錯誤：如果您的應用程式使用 Apache Beam，但沒有正確的相依性或相依性版本，就會發生此錯誤。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性：

```
<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性，應用程式會從環境相依性載入該版本，而由於版本不相符，應用程式會在執行期損毀。

如需將 Apache Beam 與 Managed Service for Apache Flink 搭配使用的詳細資訊，請參閱[搭 CloudFormation 配 Apache Flink 的受管理服務搭配使用](#)。

- 應用程式失敗，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，則叢集的消費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](#) 中的 [消費者延遲監控](#)。
- 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。

## 原因和解決方案

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。如需應用程式故障的相關資訊，請參閱[應用程式重新啟](#)。
- 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相關資訊，請參閱[解決效能問題](#)。
- 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相關資訊，請參閱在[Apache Flink 的受管理服務中實作應用程式調整](#)。
- 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[解決效能問題](#)。

## 无界状态增长

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

這種情況可能有下列徵狀：

- 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。

## 原因和解決方案

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 文件中的[狀態存留時間 \(TTL\)](#)。

- 您正在執行的應用程式相依於 Apache Beam 2.25.0 版或更高版本。您可以通過[擴展](#)關鍵實驗和值來選擇退出讀取轉換的新版本 `use_deprecated_read`。BeamApplicationProperties 如需詳細資訊，請參閱 [Apache Beam 文件](#)。

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長時段內使用彙總。除 [AggregateFunctions](#) 非使用允許增量聚合，否則 Flink 需要將整個窗口的事件保持在狀態。

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，[state time-to-live](#) 可用於根據處理時間自動將資料逾時。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 RocksDB 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 Sagemaker 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式變慢。

此外，對於具有 I/O 綁定運算符的應用程序，也可以增加 Apache Flink 應用程序的託管服務的 [ParallelismPerKPU](#) 設置。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。Apache Flink 默認配置為每

次GetRecords調用 10,000 條記錄 ( 默認情況下每 200 毫秒進行一次調用 ) ，儘管每個碎片的限制僅為 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。

您可以通過檢查 CloudWatch ReadProvisionedThroughputExceeded指標並查看此指標大於零的長時間或持續時間來確認這一點。

您也可以透過觀察持續LimitExceededException錯誤，在適用於 Apache Flink 應用程式的 Amazon 受管服務的 CloudWatch 日誌中看到這一點。

解決方案：您可以執行下列其中一項作業來解決此案例：

- 降低每GetRecords次呼叫擷取的記錄數目的預設限制
- 在您的 Amazon 受管服務中為 Apache Flink 應用程式啟用自適應性讀取。如需適性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。



- 承認-這通常是一個短暫的階段，只是向發送確認並執行任何提交消息（例如 JobManager，使用卡夫卡接收器）的子任務。

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。在此階段 `snapshotState()` 被調用用於實現 `snapshotState` 接口的類；這可以是用戶代碼，因此線程轉儲對於調查這一點很有用。

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。

| SubTasks: |                     |                        |                        |                |                            |                            |                    |             |                      |  |
|-----------|---------------------|------------------------|------------------------|----------------|----------------------------|----------------------------|--------------------|-------------|----------------------|--|
|           | End to End Duration | Checkpointed Data Size | Sync Duration          | Async Duration | Processed (persisted) Data | Alignment Duration         | Start Delay        |             |                      |  |
| Minimum   | 495ms               | 11.1 KB                | 8ms                    | 357ms          | 0 B (0 B)                  | 0ms                        | 126ms              |             |                      |  |
| Average   | 813ms               | 586 KB                 | 28ms                   | 653ms          | 0 B (0 B)                  | 0ms                        | 126ms              |             |                      |  |
| Maximum   | 1s                  | 1.70 MB                | 69ms                   | 1s             | 0 B (0 B)                  | 1ms                        | 128ms              |             |                      |  |
| ID        | Acknowledged        | End to End Duration    | Checkpointed Data Size | Sync Duration  | Async Duration             | Processed (persisted) Data | Alignment Duration | Start Delay | Unaligned Checkpoint |  |
| 0         | 2022-03-02 14:16:49 | 566ms                  | 11.1 KB                | 8ms            | 429ms                      | 0 B (0 B)                  | 0ms                | 126ms       | false                |  |
| 1         | 2022-03-02 14:16:50 | 1s                     | 1.70 MB                | 69ms           | 1s                         | 0 B (0 B)                  | 0ms                | 128ms       | false                |  |
| 2         | 2022-03-02 14:16:49 | 495ms                  | 11.1 KB                | 8ms            | 357ms                      | 0 B (0 B)                  | 1ms                | 126ms       | false                |  |

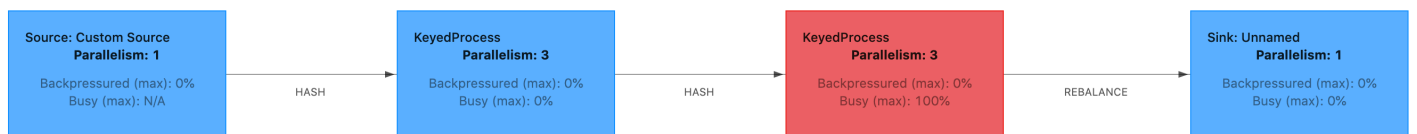
  

|                 |            |                     |       |     |           |
|-----------------|------------|---------------------|-------|-----|-----------|
| - Sink: Unnamed | 1/1 (100%) | 2022-03-02 14:16:49 | 131ms | 0 B | 0 B (0 B) |
|-----------------|------------|---------------------|-------|-----|-----------|

| SubTasks: |  |  |  |  |  |  |  |  |  |  |
|-----------|--|--|--|--|--|--|--|--|--|--|
|-----------|--|--|--|--|--|--|--|--|--|--|

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。以下是一個例子，JobGraph 其中第二個 KeyedProcess 操作員是忙。



您可以通過使用 Flink 火焰圖或 TaskManager 線程轉儲來調查需要很長時間的時間。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 接受執行 JVM 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(KeyedProcessOperator
at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStreamTask
at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTask
at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProcessor
...
```

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。操作符線程可以通過它們的名稱來標識，因為操作符線程與操作符具有相同的名稱，以及它與哪個子任務相關的指示，例如，KeyedProcess ( 1/3 ) #0 線程來自KeyedProcess操作符並且來自第一個 ( 3 個 ) 子任務。

執行緒可以是下列幾種狀態之一：

- NEW— 執行緒已建立，但尚未處理
- RUNNABLE— 執行緒在 CPU
- BLOCKED-線程正在等待另一個線程釋放它的鎖
- WAITING— 執行緒正在使用wait()join()、或park()方法等待
- TIMED\_WAITING — 線程通過使用睡眠，等待，加入或停放方法等待，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)。

要確定選擇特定運算符時可以使用選 TaskManagers 項卡的線程轉儲的任務管理器。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員上執行。

The screenshot displays the Flink Web UI interface. On the left, a diagram shows a red box labeled 'KeyedProcess' with 'Parallelism: 3'. Below it, it indicates 'Backpressured (max): 0%' and 'Busy (max): 100%'. A dashed blue arrow labeled 'HASH' points into the box from the left, and another labeled 'REBALANCE' points out to the right. On the right side, a table lists TaskManagers with columns for Host, LOG, Bytes received, Records received, Bytes sent, Records sent, and Status. Two TaskManagers are shown, both with a 'RUNNING' status.

| Host                        | LOG | Bytes received | Records received | Bytes sent | Records sent | Status  |
|-----------------------------|-----|----------------|------------------|------------|--------------|---------|
| ip-142-151-131-22:61<br>21  | LOG | 936 B          | 0                | 0 B        | 0            | RUNNING |
| ip-142-151-146-195:6<br>121 | LOG | 103 KB         | 1,423            | 71.1 KB    | 1,422        | RUNNING |

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，以下堆棧跟踪來自 KeyedProcess 操作符，並且是第一個子任務。

```
"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(KeyedProcessOperator.java:100)
at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStreamTaskNetworkInput.java:100)
at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTaskNetworkInput.java:100)
at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProcessor.java:100)
...
```

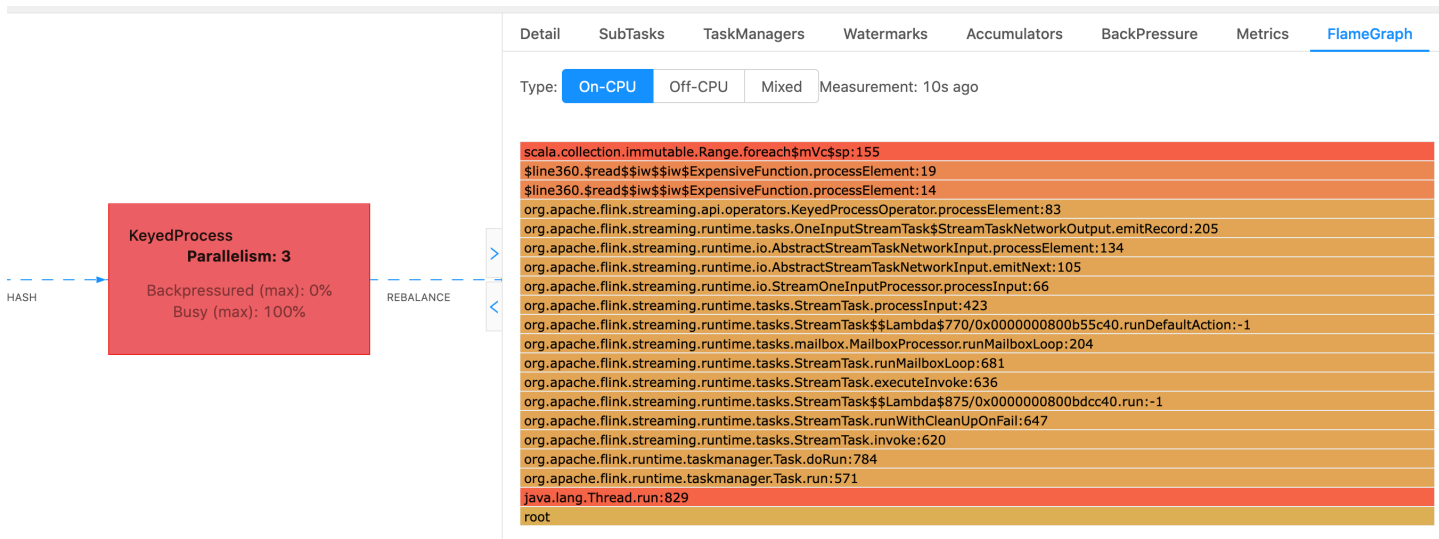
如果有多個運算子具有相同名稱，則可能會造成混淆，但我們可以透過命名運算子來解決這個問題。例如：

```
....
.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 FlameGraph 索引標籤。一旦收集到足夠的樣本，火焰圖即會顯示。以下是 FlameGraph 為了採取 ProcessFunction 了大量的時間檢查點。



這是一個非常簡單的火焰圖表，並表明所有的CPU時間都在 `ExpensiveFunction` 運營商的 `foreach` 外觀內 `processElement` 度過。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詳細資訊，請參閱 Apache Flink 文件中的 [資料類型和序列化](#)。
- 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`

如需監視檢查點的詳細資訊，請參閱 Apache Flink 文件中的[監視檢查點](#)。

## 原因和解決方案

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：

- 您的應用程式KPU佈建不足。如需增加應用程式佈建的相關資訊，請參閱[在 Apache Flink 的受管理服務中實作應用程式調整](#)。
- 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

如果您的 Beam 應用程式[shutdownSourcesAfterIdleMs](#)設定為 0ms，則檢查點可能無法觸發，因為工作處於 "FINISHED" 狀態。本節說明此狀況的徵狀和解決方案。

### 徵狀

移至 Apache Flink 應用程式 CloudWatch 記錄檔的受管理服務，並檢查是否已記錄下列記錄訊息。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(CheckpointCoordinator)",
  "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，其中某些任務已進入 "FINISHED" 狀態，並且不再可能執行檢查點。

| Detail | SubTasks       | TaskManagers     | Watermarks | Accumulators | BackPressure | Metrics             | FlameGraph          |          |          |      |
|--------|----------------|------------------|------------|--------------|--------------|---------------------|---------------------|----------|----------|------|
| ID     | Bytes Received | Records Received | Bytes Sent | Records Sent | Attempt      | Host                | Start Time          | Duration | Status   | More |
| 0      | 0 B            | 0                | 0 B        | 0            | 1            | sea3-ws-agg-r3-pc-1 | 2022-06-06 11:16:03 | 13m 57s  | RUNNING  | ...  |
| 1      | 0 B            | 0                | 0 B        | 0            | 1            | sea3-ws-agg-r3-pc-1 | 2022-06-06 11:16:03 | 3s       | FINISHED | ...  |
| 2      | 0 B            | 0                | 0 B        | 0            | 1            | sea3-ws-agg-r3-pc-1 | 2022-06-06 11:16:03 | 3s       | FINISHED | ...  |
| 3      | 0 B            | 0                | 0 B        | 0            | 1            | sea3-ws-agg-r3-pc-1 | 2022-06-06 11:16:03 | 3s       | FINISHED | ...  |
| 4      | 0 B            | 0                | 0 B        | 0            | 1            | sea3-ws-agg-r3-pc-1 | 2022-06-06 11:16:03 | 3s       | FINISHED | ...  |

## 原因

shutdownSourcesAfterIdleMs 是 Beam 組態變數，可關閉閒置時間 (以毫秒為單位) 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](#)。

工作進入 "FINISHED" 狀態的其中一個原因 shutdownSourcesAfterIdleMs 是何時設定為 0ms，這表示閒置的工作會立即關閉。

## 解決方案

若要防止工作立即進入 FINISHED "" 狀態，shutdownSourcesAfterIdleMs 請設定為「長」。MAX\_VALUE. 這可以透過兩種方式進行：

- 選項 1：如果您的樑組態是在 Apache Flink 應用程式的受管理服務組態頁面中設定，則您可以新增新的金鑰值對來設定 shutdownSourcesAfterIdleMs，如下所示：

| Group                     | Key                        | Value               |
|---------------------------|----------------------------|---------------------|
| BeamApplicationProperties | ShutdownSourcesAfterIdleMs | 9223372036854775807 |

- 選項 2：如果在 JAR 檔案中設定了樑規劃，則可以按 shutdownSourcesAfterIdleMs 如下方式進行設定：

```
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。該操作可能需要的CPU資源可能超過操作員可用的資源，操作員可能會等待 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。
- 這是一個資源密集型操作 ( 如果沒有數據偏斜，請考慮擴展為CPU/內存綁定工作或增加 ParallelismPerKPU I/O 綁定工作 )
- 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吞吐量

[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散式執行。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/發送的字任務 (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 (如視窗) 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，對於工作階段視窗，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。

# 阿帕奇 Flink 的 Amazon 託管服務的文檔歷史記錄

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 <a href="#">什麼是 Managed Service for Apache Flink ?</a>  | 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 <a href="#">為 Apache Flink 應用程式建立受管理服務</a> 。 | 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 <a href="#">Flink 1.13.2 入門</a> 。            | 2021 年 10 月 13 日 |

| 變更                     | 描述   | 日期               |
|------------------------|--|------------------|
| 支援 Python              | Managed Service for Apache Flink 現支援搭配使用 Python 與 Apache Flink 資料表 API & SQL 的應用程式。如需詳細資訊，請參閱 <a href="#">使用 Python 與阿帕奇 Flink 的託管服務</a> 。                                       | 2021 年 3 月 25 日  |
| 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 <a href="#">為 Apache Flink 應用程式建立受管理服務</a> 。 | 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 <a href="#">搭配 Apache Flink 管理服務使用 Apache Flink 儀表板</a> 。   | 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 <a href="#">增強型扇出 (EFO) 取用者</a> 。  | 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 <a href="#">搭 CloudFormation 配 Apache Flink 的受管理服務搭配使用</a> 。   | 2020 年 9 月 15 日  |

| 變更                      | 描述   | 日期              |
|-------------------------|--|-----------------|
| 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 <a href="#">???</a> 。                             | 2020 年 7 月 21 日 |
| 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 <a href="#">自訂信任存放區</a> 。                    | 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 <a href="#">???</a> 。              | 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 <a href="#">???</a> 。      | 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 <a href="#">???</a> 。                   | 2020 年 5 月 12 日 |
| 範例：從不同帳戶的 Kinesis 串流中讀取 | 了解如何在適用於 Apache Flink 的受管理服務應用程式中，以不同 AWS 帳戶存取 Kinesis 串流。如需詳細資訊，請參閱 <a href="#">跨帳戶</a> 。 | 2020 年 3 月 30 日 |

| 變更   | 描述  | 日期               |
|--|---|------------------|
| 支援 Apache Flink 1.8.2                      | Managed Service for Apache Flink 現支援使用 Apache Flink 1.8.2 版的應用程式。使用 Flink StreamingFileSink 連接器將輸出直接寫入 S3。如需詳細資訊，請參閱 <a href="#">為 Apache Flink 應用程式建立受管理服務</a> 。 | 2019 年 12 月 17 日 |
| 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 <a href="#">為 Apache Flink 設定受管服務，以存取 Amazon 中的資源 VPC</a> 。                                    | 2019 年 11 月 25 日 |
| Managed Service for Apache Flink 最佳實務      | Managed Service for Apache Flink 應用程式建立和管理最佳實務。如需詳細資訊，請參閱 <a href="#">???</a> 。   | 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 <a href="#">???</a> 。  | 2019 年 6 月 26 日  |
| Managed Service for Apache Flink 應用程式執行期屬性 | 使用 Managed Service for Apache Flink 中的執行期屬性。如需詳細資訊，請參閱 <a href="#">在 Apache Flink 的受管理服務中使用執行階段屬性</a> 。   | 2019 年 6 月 24 日  |

| 變更                                       | 描述   | 日期               |
|--|--|------------------|
| 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 <a href="#">將標籤新增至 Apache Flink 應用程式的受管理服務</a> 。                           | 2019 年 5 月 8 日   |
| 記錄與阿帕奇 Flink API 調用的託管服務 AWS CloudTrail  | 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。如需詳細資訊，請參閱 <a href="#">???</a> 。 | 2019 年 3 月 22 日  |
| 建立應用程式 (Firehose 收器)                     | 練習為 Apache Flink 建立受管服務，並以 Amazon Kinesis 資料串流做為來源，並將 Amazon 資料 Firehose 串流做為接收器。如需詳細資訊，請參閱 <a href="#">Firehose 水槽</a> 。  | 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 |



# 適用於阿帕奇 Flink API 範例程式碼的託管服務

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 做為 AWS Command Line Interface (AWS CLI) 動作的輸入，請將要求儲存在 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 kinesisanalyticstv2 start-application --cli-input-json file://start.json
```

如需有關搭配使用 JSON 的詳細資訊 AWS CLI，請參閱《使用AWS Command Line Interface 者指南》中的 < [產生 CLI 架構和 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

下列[AddApplicationCloudWatchLoggingOption](#)動作要求程式碼範例會將 Amazon CloudWatch 記錄選項新增至 Apache Flink 應用程式的受管服務：

```
{
  "ApplicationName": "MyApplication",
  "CloudWatchLoggingOption": {
    "LogStreamARN": "arn:aws:logs:us-east-1:123456789123:log-group:my-log-group:log-stream:My-LogStream"
  },
  "CurrentApplicationVersionId": 2
}
```

## AddApplicationInput

下列[AddApplicationInput](#)動作的要求程式碼範例會將應用程式輸入新增至 Apache Flink 應用程式的受管理服務：

```
{
  "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

下列[AddApplicationInputProcessingConfiguration](#)動作的要求程式碼範例會將應用程式輸入處理組態新增至 Apache Flink 應用程式的受管理服務：

```

{
    "ApplicationName": "MyApplication",
    "CurrentApplicationVersionId": 2,
    "InputId": "2.1",
    "InputProcessingConfiguration": {
        "InputLambdaProcessor": {
            "ResourceARN": "arn:aws:lambda:us-
east-1:012345678901:function:MyLambdaFunction"
        }
    }
}

```

## AddApplicationOutput

下列[AddApplicationOutput](#)動作的要求程式碼範例會將 Kinesis 資料串流做為應用程式輸出新增至 Apache Flink 應用程式的受管理服務：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 2,
  "Output": {
    "DestinationSchema": {
      "RecordFormatType": "JSON"
    },
    "KinesisStreamsOutput": {
      "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleOutputStream"
    },
    "Name": "DESTINATION_SQL_STREAM"
  }
}
```

## AddApplicationReferenceDataSource

下列[AddApplicationReferenceDataSource](#)動作的要求程式碼範例會將 CSV 應用程式參考資料來源新增至 Apache Flink 應用程式的受管理服務：

```
{
  "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

下列[AddApplicationVpcConfiguration](#)動作的要求程式碼範例會將 VPC 組態新增至現有應用程式：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 9,
  "VpcConfiguration": {
    "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
    "SubnetIds": [ "subnet-0123456789abcdef0" ]
  }
}
```

## CreateApplication

下列[CreateApplication](#)動作的範例要求程式碼會為 Apache Flink 應用程式建立受管理的服務：

```
{
  "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

下列[CreateApplicationSnapshot](#)動作的要求程式碼範例會建立應用程式狀態的快照集：

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

## DeleteApplication

下列[DeleteApplication](#)動作的要求程式碼範例會刪除 Apache Flink 應用程式的受管理服務：

```
{"ApplicationName": "MyApplication",
 "CreateTimestamp": 12345678912}
```

## DeleteApplicationCloudWatchLoggingOption

下列[DeleteApplicationCloudWatchLoggingOption](#)動作請求程式碼範例會從 Apache Flink 應用程式的受管服務中刪除 Amazon CloudWatch 記錄選項：

```
{
  "ApplicationName": "MyApplication",
  "CloudWatchLoggingOptionId": "3.1"
  "CurrentApplicationVersionId": 3
}
```

## DeleteApplicationInputProcessingConfiguration

下列[DeleteApplicationInputProcessingConfiguration](#)動作的要求程式碼範例會從 Apache Flink 應用程式的受管理服務移除輸入處理組態：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "InputId": "2.1"
}
```

## DeleteApplicationOutput

下列[DeleteApplicationOutput](#)動作的要求程式碼範例會從 Apache Flink 應用程式的受管理服務移除應用程式輸出：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "OutputId": "4.1"
}
```

## DeleteApplicationReferenceDataSource

下列[DeleteApplicationReferenceDataSource](#)動作的要求程式碼範例會從 Apache Flink 應用程式的受管理服務中移除應用程式參考資料來源：

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 5,
  "ReferenceId": "5.1"
}
```

## DeleteApplicationSnapshot

下列[DeleteApplicationSnapshot](#)動作的要求程式碼範例會刪除應用程式狀態的快照集：

```
{
  "ApplicationName": "MyApplication",
  "SnapshotCreationTimestamp": 12345678912,
  "SnapshotName": "MySnapshot"
}
```

## DeleteApplicationVpcConfiguration

下列[DeleteApplicationVpcConfiguration](#)動作的範例要求程式碼會從應用程式移除現有的 VPC 組態：

```
{
```



```
"ApplicationName": "MyApplication",
"CurrentApplicationVersionId": 9,
"VpcConfigurationId": "1.1"
}
```

## DescribeApplication

下列[DescribeApplication](#)動作的要求程式碼範例會傳回有關 Apache Flink 應用程式之受管理服務的詳細資訊：

```
{"ApplicationName": "MyApplication"}
```

## DescribeApplicationSnapshot

下列[DescribeApplicationSnapshot](#)動作的要求程式碼範例會傳回有關應用程式狀態快照的詳細資訊：

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

## DiscoverInputSchema

下列[DiscoverInputSchema](#)動作的範例要求程式碼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"
}
```

下列[DiscoverInputSchema](#)動作的範例要求程式碼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

下列[ListApplications](#)動作的要求程式碼範例會傳回您帳戶中 Apache Flink 應用程式的受管理服務清單：

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

## ListApplicationSnapshots

下列[ListApplicationSnapshots](#)動作的要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

下列動作的要求程式碼範例會啟[StartApplication](#)動 Apache Flink 應用程式的受管理服務，並從最新的快照集 (如果有的話) 載入應用程式狀態：

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

## StopApplication

下列 [API\\_StopApplication](#) 動作的範例要求程式碼會停止 Apache Flink 應用程式的受管理服務：

```
{"ApplicationName": "MyApplication"}
```

## UpdateApplication

下列 [UpdateApplication](#) 動作要求程式碼範例會更新 Apache Flink 應用程式的受管理服務，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](#)。

此內容已移至發行版本。請參閱[發行版本](#)。